Solving Multiple Timetabling Problems at Danish High Schools

Kristiansen, Simon

Publication date:
2014

Document Version
Peer reviewed version

Ph.D. thesis

Solving Multiple Timetabling Problems at Danish High Schools

Simon Kristiansen

November 28, 2013

Dansk titel:
Løsning af flere planlægningsproblemer på de danske gymnasier
Type: Ph.D.-afhandling

Forfatter: Simon Kristiansen

ISBN-nr:

Division of Management Science
Department of Management Engineering
Technical University of Denmark
Produktionstorvet, Building 426,
DK-2800 Kgs. Lyngby, Denmark
Phone: +45 45 25 48 00, Fax: +45 45 25 48 05
phd@man.dtu.dk

MaCom A/S
Vesterbrogade 48, 1.
DK-1620 Kbh V., Denmark
Phone: +45 33 79 79 00

November, 2013

Abstract

Planning problems at educational institutions are often time-consuming and complex tasks. Educational planning problems are studied using operational research techniques, which have been used with success and resulting in great improvements on the field. Educational planning problems are often divided into four main categories; University Course Timetabling, High School Timetabling, Examination Timetabling and Student Sectioning.

This Ph.D. thesis addresses some of the planning problems the high schools are struggling with annually, and it is partitioned into three research directions; High School Timetabling, Student Sectioning and the Meeting Planning Problem.

The underlying work of this thesis is carried out as an Industrial Ph.D. project in co-operation with the Danish software company MaCom A/S, which delivers administrative software solutions for high schools in Denmark.

Research in High School Timetabling has mainly been concentrated on local problems until recently. By the creation of an XML-format, XHSTT, and applicable wide ranging benchmark instances, it is been possible to solve the more generalized High School Timetabling problem. The first part of this thesis presents two approaches for the High School Timetabling problem of XHSTT; a heuristic method and an exact method. The heuristic method presented is an Adaptive large Neighborhood Search (ALNS) and was submitted as a contribution to the Third International Timetabling Competition in 2011. The algorithm was one of the finalists, and it achieved a third place. For the exact method a Mixed Integer Programming (MIP) model has been developed, and a two stage method has been used to solve it in a state-of-the-art MIP solver. Using the exact method has made it possible to generate lower bounds for several instances and to prove optimality of a few.

In the second part two different Student Sectioning problems at Danish high schools have been presented with three papers. Firstly the Elective Course Planning problem has been presented, which is the problem of assigning 2nd and 3rd year students to elective courses given their requests. The problem has been solved using Dantzig-Wolfe decomposition in a Branch-and-Bound framework. The method applied shows that a previously applied method performs poorly and is insufficient. A more comprehensive model has been created, containing the lacking constraints, using the more appropriate name; Elective Course Student Sectioning. The problem is solved using ALNS and solutions are proven to be close to optimum. The algorithm has been implemented and made available for the majority of the high schools in Denmark.

The second Student Sectioning problem presented is the sectioning of each first year student to a cohort, based on his/hers study line request, and two elective courses based on the requests for these. The High School Student Sectioning problem has been modeled as a bipartite network model and solved using two solution methods, a direct and a sequential method. The results show that by using a sequential method it is possible to gain much better results.

The last part of the thesis is concerned the Meeting Planning Problem and presented with two papers.

The Consultation Timetabling Problem is one of the minor, but still time-consuming, planning problems at the Danish high schools. Two types of consultations are presented; the Parental Consultation Timetabling Problem (PCTP) and the Supervisor Consultation Timetabling Problem (SCTP). One mathematical model containing both consultation types has been created and solved using an ALNS approach. The received solutions are close to optimum.

Based on the Consultation Timetabling Problem, a Generalized Meeting Planning Problem (GMPP) has been developed. Column generation with a Branch-and-Price (B&P) algorithm has been applied as solution method for the GMPP, and it has been tested on the PCTP and SCTP problem instances. The results prove that the B&P algorithm is an efficient method for solving the GMPP, and that the ALSN method performs better than anticipated.

The thesis has been aiming at solving the planning problems to optimality, or near optimality, and opening up research within the area of educational timetabling. It gives a thorough introductions to the domain of educational planning problems and presents several solution methods for High School Timetabling, Student Sectioning and the Meeting Planning Problem.

From an industrial point of view, this thesis has been able to formulate different high school planning problems as mathematical models and solve them using operational research techniques. Two of the models and the suggested solution methods have resulted in implementations in an actual decision support software, and are hence available for the majority of the high schools in Denmark. These implementations can improve the solution process in terms of time and quality.

Resumé

(Summary in Danish)

Planlægningsproblemer for uddannelsesinstitutioner er ofte både tidskrævende og komplekse. Uddannelsesrelaterede skemalægningsproblemer er et forskningsområde indenfor operationsanalyse, hvor teknikker er blevet anvendt med succes og har resulteret i store forbedringer indenfor området. Planlægningsproblemer indenfor undervisningssektoren er ofte opdelt i fire hovedområder; Universitets skemalægning, gymnasial skemalægning, eksamensplanlægning og elevfordeling.

Denne ph.d. afhandling adresserer nogle af de planlægningsproblemer som gymnasierne kæmper med, og er opdelt i tre områder; første del omfatter skemalægningen på gymnasier, anden del har fokus på holdpakning af eleverne og den sidste del omhandler mødeplanlægningsproblemer.

Det underliggende arbejde af denne afhandling er udført som et ErhvervsPhD-projekt i samarbejde med den danske software-virksomhed MaCom A/S, der leverer administrative softwareløsninger til gymnasierne i Danmark.

Forskning af gymnasial skemalægning har hovedsageligt været koncentreret om enkelte lokale gymnasier. Med oprettelsen af et XML-format, XHSTT, med dertilhørende omfattende benchmarking datasæt, er der blevet skabt en mere generaliseret problemstilling. Denne afhandling præsenterer to forskellige løsningsmetoder for XHSTT skemalægningsproblemet; en heuristisk metode og en eksakt metode. Adaptive Large Neighborhood Search (ALNS) er anvendt til den heuristiske metode og var et bidrag til den tredje internationale skemalægningskonkurrence. Algoritmen var blandt finalisterne og opnåede at få en tredjeplads.

Til den eksakte metode er der anvendt heltalsprogrammering til at lave en matematisk model til problemet og en to-trins løsningsmetode er anvendt i et avanceret løsningsværktøj. Ved brug af den eksakte metode har det været muligt at generere nogle grænseværdier for flere af problemerne, og bevise optimalitet i nogle enkelte tilfælde.

I den anden del af afhandlingen er to forskellige holdpakningsproblemer på de danske gymnasier præsenteret i tre artikler. Først er holdpakningsproblemet for 2. og 3. års elever præsenteret. Problemet består i at tildele eleverne til valgfagsklasser baseret på deres respektive ønsker. Problemet er blevet løst ved brug af Dantzig-Wolfe dekomponering i en Branch-and-Price (B&P) konstruktion. Resultaterne viser at en tidligere anvendt metode performer dårligt og generelt er utilstrækkelig. En mere omfattende matematisk model er oprettet og løst ved brug af ALNS og løsningerne har vist at være tæt på de optimale.

Det andet holdpakningproblem er førsteårsholdpakning, hvor de nye studerende skal tildeles en stamklasse baseret på deres retningslinjer og to valgfagsklasser, baseret på valgene for disse. Førsteårs holdpakningen er blevet udformet som et todelt netværksproblem og løst ved brug af to løsningsmetoder, en direkte og en sekventiel. Resultaterne viser, at man ved brug af en sekventiel metode opnår langt bedre resultater.

Den sidste del af afhandlingen omhandler mødeplanlægningsproblemet og er præsenteret i to artikler. Konsultationsproblemet er et af de mindre, men stadig tidskrævende, planlægningsproblemer på de danske gymnasier. Der findes to typer konsultationer på de danske gymnaeier; forældrekonsultationer og vejlederkonsultationer. En matematisk model omfattende begge typer er etableret og løst ved brug ALNS. De opnåede resultater er tæt på at være optimale.

Baseret på konsultationsproblemet er der blevet udviklet et mere generaliseret mødeplanlægningsproblem. Kolonnegenerering og en B&P algoritme er anvendt på det generaliserede problem og testet på de to konsultationstyper. Resultaterne viser, at den udviklede B&P algoritme er en effektiv løsningsmetode og at ALNS metoden performer bedre end først antaget.

Denne afhandling satser samlet set på at løse de nævnte planlægningsproblemer til optimalitet, eller nær-optimalitet, og at åbne forskningsfeltet indenfor uddannelsesrelaterede planlægningsproblemer.

Den giver en grundig introduktion til domænet af uddannelsesrelaterede planlægningsproblemer samt en præsentation af flere løsningsværktøjer til at løse problemer som gymnasieskemalægning, holdpakning og mødeplanlægning.

Fra et industrielt synspunkt, har det været muligt at formulere forskellige planlægningsproblemer som matematiske modeller og løse dem ved brug af teknikker indenfor operationsanalyse. To af modellerne, og de dertilhørende foreslåede løsningsmetoder, har resulteret i implementeringer i et aktuelt beslutningsværktøj, og er dermed anvendelig for størstedelen af de danske gymnasier. Disse implementeringer har forbedret planlægningsprocessen i form of reduceret arbejdstid og øget løsningskvalitet.

Preface

The work presented in this dissertation fulfills the requirements for acquiring the degree *Philosophiae Doctor* (Ph.D.) at Technical University of Denmark, as well as the *Industrial Ph.D. diploma* awarded by the Danish Ministry of Science, Technology and Innovation for Ph.D. students following the Industrial Ph.D. program.

The Ph.D. study was performed at DTU Management Engineering, Technical University of Denmark, from December 2010 to December 2013.

As part of the Industrial Ph.D. program, I have been employed at the privately held company MaCom A/S in Copenhagen. Here I have spent half of my time during the Ph.D. and worked as an integrated part of the development section, where all the results herein have been implemented into the product of MaCom A/S, Lectio. Currently the work of two papers of this thesis has resulted in operational software usable for the majority of Danish high schools.

Lecturer Thomas R. Stidsen, DTU Management Engineering has supervised the study, and Development Manager Michael B. Herold, MaCom A/S, has co-supervised the project on behalf of MaCom A/S.

This thesis is part of a larger research project concerning planning problems at educational institutions. The overall research project contains an additional Ph.D. project carried out by Matias Sørensen. The two Ph.D. projects are closely connected and have led to some collaboration.

The thesis consists of eight research papers. Two of these papers are published and one accepted for publication in scientific journal, one appears in a conference proceeding, one is published as a technical report, and the remaining three are submitted to peer review journals within the field. All papers are co-authored.

Copenhagen, Denmark, November 2013

Simon Kristiansen

Acknowledgements

First of all I would like to thank my supervisors, Thomas R. Stidsen, from DTU Management, and Michael B. Herold, from MaCom A/S, for their excellent guidance throughout the entire project. Thomas for his great supervision, his encouragement and open mind towards new ideas. Without the support of Michael, this project would never have had the same level of real world impact, and it would not have contributed with running decision support software. I would also thank David Pisinger for his happy spirits and for always being helpful when needed.

Great thanks to Senior Lecturer Andrew Mason for inviting me to visit the University of Auckland. It was a great experience to work with Andrew and I thank him for his commitment to my projects. It should be mentioned that my stay abroad was partially supported by the European Union Seventh Framework Program (FP7-PEOPLE-2009-IRSES) under grant agreement number 246647 and by the New Zealand Government as part of the OptALI project. And I am very grateful of getting this financial support, which made it possible for me to stay in Auckland for six month.

Thanks to all the co-authors and for their contribution in the different projects I have been involved in. A special thanks to Matias Sørensen, whom I have shared office with a both DTU and MaCom A/S, for close collaboration and tremendous support throughout my Ph.D.. I highly appreciate the teamwork and discussions.

I would like to thank my colleagues at DTU Management Engineering as well as my colleagues at MaCom A/S for fruitful discussions, interest in my project and a pleasant work environment. It has been a pleasure to be part of them both.

Thanks to Kia Kristiansen, Peter Vest Hansen, Simon Bull and Tor Justesen for proofreading and commenting on parts of this thesis.

Finally, great thanks goes to my family and friends, and to my wonderful girlfriend Maria Cloetta Jensen for her encouragements and support, and for letting me spend six month away when I was in Auckland. Thanks!

List of Papers

(Chronological order)

- **Article:** Simon Kristiansen, Matias Sørensen and Thomas R. Stidsen (2011). "Elective Course Planning".
  In: *European Journal of Operational Research.* (Published)
- **Conference Abstract:** Matias Sørensen, Simon Kristiansen and Thomas R. Stidsen (2012). "International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm".
  In: *Conference proceedings of PATAT 2012.* (Published)
- **Article:** Simon Kristiansen, Matias Sørensen, Michael B. Herold and Thomas R. Stidsen (2013). "The Consultation Timetabling Problem at Danish High Schools".
  In: *Journal of Heuristics.* (Published)
- **Article:** Simon Kristiansen and Thomas R. Stidsen (2013). "Elective Course Student Sectioning at Danish High Schools".
  In: Special issue of *Annals of Operations Research* in collaboration with *PATAT2012.* (Accepted with minor revision)
- **Article:** Simon Kristiansen, Matias Sørensen and Thomas R. Stidsen (2013). "Integer Programming for the Generalized (High) School Timetabling Problem".
  In: *Journal of Scheduling.* (Submitted)
- **Article:** Niels-Christian F. Bagger, Matias Sørensen, Simon Kristiansen and Thomas R. Stidsen (2013). "A Branch & Price Algorithm for the Generalized Meeting Planning Problem".
  In: *Computers and Operations Research.* (Submitted)
- **Article:** Simon Kristiansen, Thomas R. Stidsen and Andrew Mason (2013). "High School Student Sectioning at Danish High Schools".
  In: *European Journal of Operational Research.* (Submitted)
- **Technical Report:** Simon Kristiansen and Thomas R. Stidsen (2013). "A Comprehensive Study of Educational Timetabling - a survey".
  In: *DTU Mananagement Technical Report.* (Published)

Contents

Part I

Introduction

Chapter 1

Introduction and Thesis Motivation

All over the world, educational institutions are struggling with diverse student administrative planning problems. These planning problems are all personnel allocation problems. The main issue of the problems is to provide the students and teachers with satisfactory timetables. However, the school administrations also have to consider educational and economic issues when planning the timetables. Researchers and commercial (local or external) software vendors are still working on these problems, as they can vary a lot from institution to institution and from country to country. In the literature, educational planning problems are often divided into four main categories; *University Course Timetabling, High School Timetabling, Examination Timetabling* and *Student Sectioning*.

University Course Timetabling: University Course Timetabling is the problem of assigning lectures of courses to time slots, rooms and possibly other resources, subject to constraints applied for the single student. The large size and structure of universities often makes the problem of solving the course timetabling so complicated that it is solved by several experts scattered across different faculties and departments. Due to the complexity of the problem, it is quite common that the universities re-use the solution from previous years.

University Course Timetabling is often divided into two general approaches; the *Curriculum-based University Course Timetabling* and the *Enrollment-based University Course Timetabling*.

Courses which can be taken in combination, because they are needed to satisfy the degree rules of the given study, are called curricula. Assigning courses to time slots based on this is called Curriculum-based University Course Timetabling (Burke et al., 2012; Lach and Lübbecke, 2012). Enrollment-based University Course Timetabling is based on the enrollment data of each individual student, and it is then determined where courses are placed in the timetable, such that all students can attend the events in which they are enrolled (Cambazard et al., 2008; Nothegger et al., 2012).

High School Timetabling: At high school the students are grouped in classes prior to the timetabling problem (using Student Sectioning). The students of a class are usually grouped together for all the mandatory courses. Essentially, the High School Timetabling problem is then to allocate these classes to time slots, teachers and rooms, to satisfy the constraints of the problem (Post et al., 2012a; Pillay, 2013).

The grouping of students to classes is one of the main differences between High School Timetabling and University Course Timetabling. Another difference is the assignment of teachers. High school teachers often teach full time, whereas in universities the professors/lecturers usually only teach one course. Hence the clashes constraints on the teachers as well as preferred teacher constraints are added to the High School Timetabling problem.

Examination Timetabling: The Examination Timetabling problem is the task of scheduling a given number of exams to a limited number of time slots. Each course has one event representing the exam. The problem is to avoid clashes in each student's examination timetable, and to make sure that they have sufficient preparation time for each exam (Qu et al., 2009; Müller, 2013).

Examination Timetabling and University Course Timetabling are the two most studied subjects of education planning problems. It is often the Examination Timetabling problem at universities which is discussed in the literature, and this is relatively close to the University Course Timetabling (Schaerf, 1999). However, it is broadly accepted to distinguish between the two problems, due to their characteristics (McCollum, 2007a). Firstly, there is only one exam for each course, where there can be several lectures of the same course in University Course Timetabling, and secondly there is variations in the constraints. In course timetabling, the universities pursue a compact timetable, whereas the Examination Timetabling is often requested to be spread out. A student cannot have two exams on successive days and the days between two exams are used as preparation time. Furthermore, there can be more than one exam in a room whereas there can only be one lecture in each room.

Student Sectioning: The previous mentioned educational planning problems are all considering the problem of assigning some events to time slots. Student Sectioning resides outside this category as this problem involves assigning students to sections and usually not times (de Haan et al., 2007; Suyanto, 2010).

If courses have too many students than it is possible to fit into one class room, the courses might be split into sections, i.e. copies of the same course with its own time slot, room and teacher. Student Sectioning is the problem of assigning students to these sections of courses while respecting the requests of the individual student. Some formulations of Student Sectioning are also trying to assign the sections to clusters or blocks. This is used to avoid student clashes in the timetable when the sections are allocated time slots.

At high schools the Student Sectioning problem is seen as a separate initial part of the High School Timetabling problem (de Haan et al., 2007; Kristiansen and Stidsen, 2013a), whereas University Course Timetabling often has student sectioning incorporated into the timetabling problem (Müller and Murray, 2010; Suyanto, 2010).

This thesis has been concentrated on High School Timetabling (Part II) and Student Sectioning (Part III). Moreover, a Master's Thesis project on Examination Timetabling has been supervised (Schadegg, 2013). Further descriptions of the mentioned educational planning problems are presented in Chapter 2.

Meeting Planning Problem: Besides High School Timetabling and Student Sectioning, a new type of planning problems has been introduced through the thesis; the *Meeting Planning Problem*. Planning meetings can become a difficult puzzle, if the occurrence of meetings is large and the meetings are interrelated.

At high schools the meeting planning problem can be used to schedule consultation meetings between parents and teachers, or meetings between the students and their supervisors (Kristiansen et al., 2013b). Rudová (2013) solves the Bachelor state examination problem which is quite similar to the supervisor consultation timetabling problem.

The meeting planning problem can be used in various cases where a large number of meetings in a crisscross pattern should be scheduled.

Part IV of this thesis is concentrated on the Meeting Planning Problem.

1.1 Motivation - the Danish High Schools

Due to a number of economic and educational reforms, managing the high schools in Denmark has become increasingly complex, and as of January 1st 2007 the Danish high schools became self-governing institutions. In 2009 an analytical report on the increased demand for IT on self-governing educational institutions in Denmark was executed (McKinsey & Company, 2009). Overall, 4,300 full-time equivalents (FTE) was used on student administration of the educational institutions and 18% of these on timetabling problems. Moreover, the report observes that the majority of the institutions use manual work during the process, either in spreadsheets or by hand. It is concluded that by improving the IT at the high schools, a total gross efficiency potential can be estimated at 346 million DKK, of which 29% comes from resource and scheduling problems.

Based on this information, a comparative analysis of timetabling planning systems for the upper secondary educations in Denmark was conducted (Capgemini, 2010). The analysis lists the pros and cons for each planning system and concludes that the existing systems are inadequate when looking at the criteria stated in McKinsey & Company (2009) as requirements for a schedule planning system.

1.2 Operations Research

Operations Research (OR) techniques will be used to study and solve the problems of this thesis. OR is a discipline within applied mathematics and computer science and is widely used on planning problems, including education timetabling problems. OR on educational timetabling has been widely used over the past 50 years (de Werra, 1985; Bardadym, 1996). Conferences and competitions have focused on the problem, and hence contributed with an increase in publications. The conferences include the *International Conference on the Practice and Theory of Automated Timetabling* (PATAT, 2013) and the *Multidisciplinary International Scheduling Conference: Theory & Application* (MISTA, 2013).

Three international timetabling competitions (ITC) on educational timetabling have been organized. The first International Timetabling Competition (ITC2003) was regarding University Course Timetabling (Paechter et al., 2002). The second, ITC2007, had three tracks. Two tracks on University Course Timetabling (one curriculum-based and one enrollment-based) and one track on Examination Timetabling (Gaspero et al., 2007; McCollum, 2007b; McCollum et al., 2010). The third and latest competition (ITC2011) was focused on the XHSTT format of High School Timetabling (Post et al., 2011, 2012b).

Many of the problems of this thesis have been proven to be $\mathcal{NP}$-hard and the following types of methods are employed for solving them:

Heuristic solution methods: Heuristics are operational research techniques which seek good solutions at a reasonable computational cost, but they do not guarantee optimality. A metaheuristic is a special case of heuristics which consists of a toplevel strategy that guides underlying heuristics to be applied to many classes of problems. In many of the recent publications some sort of hybridization of multiple heuristics are used as solution methods. These types of heuristics are denoted hyper-heuristics (Burke et al., 2003). Using hyper-heuristics makes it possible to explore a large part of the solution space and hence make it possible to solve classes of problems rather than a few.

This thesis considers the hyper-heuristic *Adaptive Large Neighborhood Search* (ALNS) for solving the High School Timetabling problem of ITC2011 (Chapter 3), the Elective Course Student Sectioning (Chapter 6) and the Consultation Timetabling Problem (Chapter 8). In ALNS, the *Large Neighborhood Search* approach is extended using multiple insertion and removal methods (Pisinger and Ropke, 2005) and has been used mainly on variants of Vehicle Routing Problems (Ropke and Pisinger, 2006; Laporte et al., 2010; Ribeiro and Laporte, 2012).

Exact solution methods: Solving a problem using an exact method makes it possible to issue certificates for optimality or the quality of the solution. Exact solution methods can either be of solving the mathematical model using a Mixed Integer Programming (MIP) solver, or by using a number of different exact method approaches such as decomposition methods, dynamic programming etc. If it is not possible to solve the problem to optimality, due to the problem instance, the properties or given restrictions, exact methods can be able to provide bounds and hence benchmark the performance of heuristics.

In this thesis, different exact methods have been applied. In Chapter 4 and 7 the XHSTT and High School Student Sectioning are solved, respectively, using direct and sequential approaches, and in Chapter 6 and 8 exact methods are used to benchmark the performance of the applied meta-heuristics by solving the mathematical model directly, using a state-of-the-art MIP solver. Chapter 5 and 9 use more advanced exact methods in term of decomposition and branching techniques (Desrosiers and Lübbecke, 2005). Chapter 5 applies Dantzig-Wolfe decomposition in a Branch-and-Price framework. Explicit Constraint Branching is used as a branching technique (Appleget and Wood, 2000). In Chapter 9, Column Generation and Branch-and-Price are used for solving the Generalized Meeting Problem.

Matheuristics: Matheuristics is a result of the combination of the two aforementioned approaches and hence *enjoy the best of both worlds* (Ryan, 2012). A matheuristic is a type of hyper-heuristic where meta-heuristics and mathematical programming (MP) techniques are embedded. By using MIP solvers or customized MIP algorithms, such as decomposition, in a heuristic context, either as primary solvers or as sub procedures (Avella et al., 2007; Maniezzo et al., 2009; Blum et al., 2011).

In this thesis, matheuristics have only been touched briefly in Chapter 6, where an MP insertion technique is applied in the ALNS framework.

1.3 Contributions

Due to the idea of an Industrial Ph.D., the motivation and contribution of this thesis are two-fold: Firstly, it is to contribute to the basic research knowledge on educational timetabling problems within high schools. Secondly, to use operational research techniques to provide good decision support tools to achieve better planning of the timetables.

1.3.1 Basic Research Contribution on Education Planning Problems

On the overall level, this Ph.D. project has contributed to the area of high school planning problems by operation research techniques and developing methods that are able to solve the problems, some implemented in administrative software. A number of promising results has been reached. Highlights of the basic research contribution of each paper of this thesis are described below:

- (Kristiansen and Stidsen, 2013b): A comprehensive survey on educational timetabling. Gives an overview of the different types of planning problems within the education sector, the available benchmarks and related literature.
- (Sørensen et al., 2012): A contribution for the Third International Timetabling Competition (ITC2011) An Adaptive Large Neighborhood Search algorithm is developed for the High School Timetabling problem.
- (Kristiansen et al., 2013c): A Mixed Integer Programming (MIP) model for High School Timetabling problems of the XHSTT format. It has been possible to provide lower bounds on several instances and prove the optimality on few.
- (Kristiansen et al., 2011): Presents the *Elective Course Planning* problem. A Dantzig-Wolfe decomposition in a Branch-and-Price framework is developed with Explicit Constraint

Branching as one of the branching techniques. The algorithm is compared with an existing solution approach and the results show that existing solution approach is insufficient in terms of lacking constraints and performance.

- (Kristiansen and Stidsen, 2013a): A MIP model and an Adaptive Large Neighborhood Search for the *Elective Course Student Sectioning* problem. The algorithm is tested on 80 real life instances from Danish high schools and provides solutions close to optimal. The algorithm is implemented in Lectio and hence available for the majority of the high schools in Denmark.

- (Kristiansen et al., 2013a): An Integer Programming model for the student scheduling problem of first year students at Danish high schools is presented. In *High School Student Sectioning*, a student should be assigned a cohort and two elective courses given his requests. Two exact solution methods are implemented, a direct and a sequential approach. The two approaches are tested on 25 real life datasets.

- (Kristiansen et al., 2013b): The *Consultation Timetabling Problem* is presented and a MIP model, containing two types of consultations, is created. Both types are of assigning students and teachers to meetings given some requests. An Adaptive Large Neighborhood Search algorithm is developed and tested on 200 real life datasets. The algorithm is implemented in Lectio and hence available for the majority of the high school in Denmark.

- (Bagger et al., 2013): A MIP model is presented for the *Generalized Meeting Planning Problem* and experimental results are conducted using Column Generation and Branch-and-Price. The algorithm is tested on the data from Kristiansen et al. (2013b) and proves that the heuristic approach provides results close to optimum.

1.3.2 Industrial Application

The role of an industrial Ph.D. is of bridging two worlds, the theoretical world of academia and the practical world of industry. Having both an academic and a practical role has given an extra dimension to the dissertation, a role I have found very challenging and rewarding.

The academic role is as part of the research team at the Operation Research section at DTU Management, and the practical role is as part of the software company MaCom A/S. MaCom A/S is a small company in Copenhagen with Lectio as their primary product. Lectio is a cloud-based high school administration system which handles all sorts of administrative tasks for Danish high schools, including the various planning problems. The majority of the high schools in Denmark are using Lectio, hence MaCom A/S is indeed aware of the challenges the high schools are facing and what they expect from the solutions to the planning problems.

The unique position at an industry partner has some advantage when doing research. Firstly is the access to knowledge and the stakeholders. Having detailed knowledge of the planning problems, and the possibility to discuss and evaluate on the problem with the stakeholders, are invaluable benefits to have when solving practical planning problems. Secondly is the access to actual data. When doing scientific research, access to real life data can be a rather difficult task, and often the data is lacking in term of both quality and quantity. The industry position can give access to data collected over several years, which in the best case are of an understanding quality.

Having the knowledge and the data of a selected problem, the challenge was to capture the problem in a mathematical model which represent the complexity. As Lectio is used by the majority of the Danish high schools it has been crucial that the models are generalized such that they cover all the high schools.

Decision Support Software for the Danish High Schools

Burke et al. (2003) write that small companies are often not interested in getting the optimal solutions for their planning problems. They are more interested in solutions which are;

good enough - soon enough - cheap enough.

This quote is a criticism of companies not aiming at solving their problem to optimality. However, when making decision support software applicable for several users, this quote is an excellent description of the way of thinking.

- *Good enough*: The term optimality is often different from the operational researcher's point of view compared to the users. From an academic research point of view a problem can be formulated as a mathematical problem and the goal is then often to solve this model to optimality (or near optimality). However, the user might not find the solution optimal as small changes might benefit the given high school, changes which are not obvious to the developer. This is due to tacit knowledge of the particular high schools. The same is applicable for the industrial role of this thesis.

  Having the majority of the Danish high schools as a focal point makes it difficult, if not impossible, to incorporate all the preferences of each single user. Hence the goal is to make a generalized mathematical model and then provide *good enough* solutions to the users. The users have the opportunity to customize the problems by changing some of the parameters.

- *Soon enough*: Soon enough denotes two things: the release date and the running time of the algorithms. Development of decision support systems is a dynamic process, where feedback from the users keeps the system up-to-date. Planning problems at educational institutions are often only solved during a short single period each year and it is therefore important that the software is accessible at this given time. At the high schools in Denmark, High School Timetabling and Student Sectioning is solved in February and March, and this adds some limits to the time-windows if feedback is wanted. When the software is available, the running time of the algorithms should be relatively short. The algorithm is part of the decision support system, and the users would often run the algorithms several times with small changes, to get exactly the outcome preferred, hence long running time is seldom an option.

- *Cheap enough*: The last issue is the price for the system. Public high schools in Denmark have a limited budget hence the software system should be fairly cheap. This limits the use of other commercial products such as state-of-the-art MIP solver as Gurobi or CPLEX for the solution approaches. Using these commercial products can increase the price on the system. The use of open-source MIP solvers or solving the problems using some sort of heuristics is often an advantage.

Having created mathematical models and appertaining solution methods which meet the above criteria is only a part of a successful practical implementation. The application needs to be wrapped in a user-friendly interface, such that the user can easily figure out how to use the program and how to make it solve their particular planning problem. In this thesis the implementation process has included the solution methods, a SQL database for data handling and a GUI in the framework of Lectio.

Furthermore, it should be remembered that creating decision support tools is an iterative process. Based on communication and feedback from the users, the conceptual models of the problems are constantly under development. It is important that the model, and hence the solutions are always tailored to the customers' needs and wants.

It has been the goal for all the projects of this thesis to provide optimal, or near optimal, solutions to generalized planning problems such that, given the inputs from the high schools, we are able to provide all the high schools with solutions of good quality within a reasonably time. Two of the proposed solution methods for the planning problems considered in this thesis have resulted in software applicable for the majority of the Danish high schools. This is the Consultation Timetabling Problem (Kristiansen et al., 2013b) and the Elective Course Student Sectioning (Kristiansen and Stidsen, 2013a). The first version of Consultation Timetabling problem was available as of May 2011, and the first version of Elective Course Student Sectioning was released in January 2012.

The other papers are more theoretical, or they remain to be integrated in a system. Chapter 7 shows that the High School Student Sectioning problem can be modeled as an Integer Programming model and solved using a state-of-the-art MIP solver. The process is missing the first evaluation of the conceptual model and the solutions, and a system implementation of it.

Chapter 3 has contributed to the company in a more indirect way. Using various Danish media, MaCom A/S has stated the solution methods of the planning problems in Lectio were developed by the finalists in the "*World Cup of High School Timetabling*"[1]. By competing in the International Timetabling Competition (ITC2011) and finishing third has resulted in good public relations for the company, as they can claim that the planning problems incorporated in Lectio have been developed by researchers compatible at an international level.

1.4 Thesis outline

This thesis is divided into five parts and consists of eight papers. Part I motivates the relevance and gives an introduction to the domain of this thesis. Part II investigates the High School Timetabling problem. In Part III different Student Sectioning problems at Danish high schools are studied and in Part IV the Meeting Planning Problem is presented. Finally, a conclusion is given in Part V.

Part I: Introduction consists of three chapters. The current chapter contains the introduction and motivation of the thesis. The introduction is relatively compact due to Chapter 2, which is a technical report containing a survey on educational timetabling.

- Chapter 2: **A Comprehensive Study of Educational Timetabling - a survey** presents a comprehensive study on the research of Educational Timetabling within the last decade. Firstly the general concepts of Educational Timetabling are presented with previous surveys and competitions. Secondly, the four major contributors of Educational Timetabling are introduced; *University Course Timetabling*, *High School Timetabling*, *Examination Timetabling* and *Student Sectioning*. For each subject a description is given together with available benchmark data and relevant literature presented within the last decade. The paper contains an appendix with all mentioned literature listed with subject and appertaining comments. The work has been presented as follows:
  - Technical report published at DTU Management (Kristiansen and Stidsen, 2013b).

Part II: High School Timetabling using the XHSTT format presents two papers on the High School Timetabling Problem. High School Timetabling has been lacking some general concepts and benchmark datasets. A group of researchers agreed on developing an XML format for the problem together with new varied benchmark data sets (Post et al., 2012a). The format was used for ITC2011. Both papers of this chapter are concerned with the solving of High School Timetabling problems of the XHSTT format.

- Chapter 3: **International Timetabling Competition 2011: An Adaptive Large Neighborhood Search Algorithm** presents a contribution to ITC2011. The created algorithm is an Adaptive Large Neighborhood Search consisting of 9 insertion methods and 14 removal methods. The final algorithm contained 9 free parameters which were tuned using the *irace package*. The algorithm received a third place and provided the best results for four instances. The work was presented at the ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT2011).

[1] `http://videnskab.dk/miljo-naturvidenskab/danskere-i-finalen-til-vm-i-skemalaegning`
`http://videnskab.dk/kultur-samfund/danskere-vinder-bronze-til-vm-i-skema`
`http://www.dtu.dk/Nyheder/2012/09/Webnyhed_DTU-vinder-VM-bronze-i-skemalaegning`
`http://gymnasieskolen.dk/vm-i-gymnasieskemaer`

  - Extended abstract for Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT), (Sørensen et al., 2012).

- Chapter 4: **Integer Programming for the Generalized (High) School Timetabling Problem** presents the first exact solution method capable of handling an arbitrary instance of the XHSTT format. The problem is formulated as a Mixed-Integer Programming (MIP) model and solved using a two stage approach in a commercial general purpose MIP solver. In the first stage a MIP model only containing the hard constraints is solved. In case of optimality Stage Two is performed. In the second stage all the soft constraints are added and the solution is warm-started with the solution from Stage One, and a constraint is added to ensure that the optimal value of the first stage is kept.

  The obvious advantage of Integer Programming over heuristic methods is the capability to issue certificates of optimality, and the primary aim of this paper is to generate good lower bounds for the different instances and hopefully prove optimality for some of them. The approach was tested on the instances of ITC2011 and the XHSTT archive `ALL_INSTANCES`. The computational results show that the suggested approach is competitive with the methods used at ITC2011. Furthermore, the results show that the approach is able to find previously unknown optimal solutions for 2 instances of `ALL_INSTANCES`, and proves optimality of 4 known solutions. For the instances not solved to optimality, new non-trivial lower bounds were found in 11 cases, and new best-known solutions were found in 9 cases. The work has been presented as follows:

  - Submitted to Journal of Scheduling (Kristiansen et al., 2013c).

Part III: Student Sectioning Problems at High Schools presents three papers concerning student sectioning problems at high schools in Denmark. The first two consider the *Elective Course Student Sectioning* (or *Elective Course Planning*) of the 2nd and 3rd year high school students, while the latter is concentrated on the first year students in *High School Student Sectioning*. For all papers the problem is of assigning students to sections of courses such that no clashes exist.

- Chapter 5: **Elective Course Planning**. In the beginning of each school year the 2nd and 3rd year students request some elective courses they would like to participate in next to their mandatory courses. The Elective Course Planning is the problem of assigning the students to course sections based on these requests while minimizing the number of classes created. A Mixed Integer Programming model for the problem is created and three different solution methods are applied; a direct MIP model solution, a MIP model solution with Explicit Constraint Branching and a Branch-and-Price algorithm with Explicit Constraint Branching. The Branch-and-Price framework is using Dantzig-Wolfe decomposition. The solution methods are tested on 98 real-life instances from Danish high schools. The results show that Explicit Constraint Branching is an interesting and efficient tool for solving this problem, and the previous applied meta-heuristics could be improved significantly. Moreover it was possible to establish optimal solutions for the majority of the instances within one hour and a more fair distribution of the students is possible. The work has been presented as follows:

  - Published in European Journal of Operational Research (Kristiansen et al., 2011).

- Chapter 6: **Elective Course Student Sectioning at Danish High Schools** presents a new MIP model for the Elective Course Planning problem, using the more appropriate name; Elective Course Student Sectioning. The model from Kristiansen et al. (2011) is improved such that it contains all necessary restriction including fairness distribution. The problem is solved using an Adaptive Large Neighborhood Search algorithm. The algorithm contains three types of removal heuristics and two types of insertion heuristics. Further three coupled heuristics are embedded. The algorithm is tested on 80 real-life instances from Denmark

and the performance is compared with solutions found using Gurobi. The results show that, given the restricted solution time, ALNS on average finds solutions within 1% of the optimum and it outperforms Gurobi on the large instances. The work has been presented as follows:

  - Accepted for publication in the special issue of *Annals of Operations Research* in collaboration with the *9th International Conference on the Practice and Theory of Automated Timetabling* (PATAT), (Kristiansen and Stidsen, 2013a).

- Chapter 7: **High School Student Sectioning at Danish High Schools** presents a different student sectioning problem at Danish high schools compared to the Elective Course Student Sectioning. When first year students are enrolled at a Danish high school, they have requested a study line and two elective courses, a linguistic course (e.g. Spanish or French) and an artistic course (e.g. Drawing or Media). The High School Student Sectioning problem is bipartite. It is of assigning the students to cohorts given their study line requests, and it is of assigning the students to elective course classes given the requests of these. A cohort is a group of students having most of their mandatory courses together. The problem is solved using a sequential approach. First the students are assigned to cohorts. Second the students are assigned to elective course classes using the fixed solution from the first step. Finally the entire problem is solved using the solutions from the first two steps as initial solution. The approach is tested on 25 instances and the results show that the approach is only 0.5% worse than the best found solution. The work has been presented as follows:

  - Submitted to European Journal of Operational Research (Kristiansen et al., 2013a).

Part IV: Meeting Planning Problem presents two papers on the fairly new planning problem within educational timetabling. The *Meeting Planning Problem* is the problem of assigning entities, such as students and teacher, to meetings given some requests. The problem was first introduced as the Consultation Timetabling Problem.

- Chapter 8: **The Consultation Timetabling Problem at Danish High Schools** is the first paper presenting the Consultation Timetabling Problem. Two types of consultation problems are presented. The first one is the *Parental Consultation Timetabling Problem* (PCTP). Twice a year the high schools in Denmark offer the possibility for school meetings, where the students (and their parents) meet the selected teachers for short meetings, to elaborate on the development of the student. The second consultation problem is the *Supervisor Consultation Timetabling Problem* (SCTP). In the last year of high schools the students are required to write a large study project and during this project it is required to have some meetings with the respective supervisors.

  The two consultation problems are modeled as one MIP model and an Adaptive Large Neighborhood Search algorithm is applied for solving it. Computational results are obtained using 300 real-life instances. The results show that the ALNS outperformed the previous applied heuristic for the PCSP. (No previous method has been applied for SCTP.) Moreover it is shown that the ALNS algorithm in average provides results within 5% from optimum for both problem types. The work has been presented as follows:

  - Published in Journal of Heuristics (Kristiansen et al., 2013b).

- Chapter 9: **A Branch & Price Algorithm for the Generalized Meeting Planning Problem** presents a more generalized model of the Meeting Planning Problem. Instead of meetings between student and teachers, the paper considers meeting between entities. A Branch-and-Price (B&P) algorithm is presented and it has been tested on two different meeting planning problems; the PCTP and the SCTP from Kristiansen et al. (2013b).

  For both problems the developed B&P algorithm is tested on 100 real-world data examples from Danish high schools. The B&P algorithm obtains an average gap of 2.32% for PCTP

and 1.15% for SCTP. These are convincing results for effectiveness of the algorithm. It is likely that there are many other problems where the Generalized Meeting Planning problem is applicable, and where the described B&P approach therefore can be applied. The work has been presented as follows:

- Submitted to Computers and Operations Research (Bagger et al., 2013).

Part V: Conclusion: The final part of the thesis contains some concluding remarks and suggestions for future work.

Bibliography

J. Appleget and R. Wood. *Explicit-Constraint Branching for Solving Mixed-Integer Programs*, chapter 14, pages 245–262. Springer Netherlands, 2000.

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

N.-C. F. Bagger, M. Sørensen, S. Kristiansen, and T. R. Stidsen. A branch & price algorithm for the generalized meeting planning problem. *Computers & Operations Research*, to appear, 2013.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

C. Blum, J. Puchinger, G. R. Raidl, and A. Roli. Hybrid metaheuristics in combinatorial optimization: A survey. *Applied Soft Computing*, 11(6):4135 – 4151, 2011. ISSN 1568-4946.

E. Burke, G. Kendall, J. Newall, E. Hart, P. Ross, and S. Schulenburg. Hyper-heuristics: An emerging direction in modern search technology. In F. Glover and G. Kochenberger, editors, *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, pages 457–474. Springer US, 2003. ISBN 978-1-4020-7263-5.

E. K. Burke, J. Marec, A. J. Parkes, and H. Rudova. A branch-and-cut procedure for the udine course timetabling problem. *Annals of Operations Research*, 194(1):71–87, 2012. ISSN 0254-5330.

H. Cambazard, E. Hebrard, B. O'Sullivan, and A. Papadopoulos. Local search and constraint programming for the post enrolment-based course timetabling problem. In *PATAT2008: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 2008.

Capgemini. uni-c - analyse af skemaplanlægningssystemer. `http://www.uvm.dk/Administration/It-og-digitalisering/Ambitioes-it/~/media/UVM/Filer/Adm/PDF12/120113%20UNIC%20Analyse%20af%20skemalaegningssystemer.ashx` [Retrieved 28/11-2013], December 2010. Comparativ analyse of timetabling systems in Denmark. Ordered by UNI-C and produced by the consulting firm Capgemini Danmark A/S. In Danish.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2): 151 – 162, 1985. ISSN 0377-2217.

J. Desrosiers and M. Lübbecke. Selected topics in column generation. *G–2002–64*, 34:1–34, 2005.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

S. Kristiansen and T. R. Stidsen. Elective course student sectioning at danish high schools. *Annals of Operations Research*, PATAT 2012 SI:To appear, 2013a.

S. Kristiansen and T. R. Stidsen. A comprehensive study of educational timetabling - a survey. Technical Report 8, 2013, DTU Management Engineering, Technical University of Denmark, November 2013b.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

S. Kristiansen, A. Mason, and T. R. Stidsen. High school student sectioning at danish high schools. *European Journal of Operational Research*, MISTA 2013:Submitted, 2013a.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013b.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Integer programming for the generalized (high) school timetabling problem. *Journal of Scheduling*, Submitted 5/9-2013, 2013c.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

V. Maniezzo, T. Stützle, and S. Voß. Matheuristics: Hybridizing metaheuristics and mathematical programming. *Annals of Information Systems*, 10, 2009.

B. McCollum. A perspective on bridging the gap between theory and practice in university timetabling. In E. Burke and H. RudovÃ¡, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23. Springer Berlin Heidelberg, 2007a. ISBN 978-3-540-77344-3. doi: 10.1007/978-3-540-77345-0_1. URL `http://dx.doi.org/10.1007/978-3-540-77345-0_1`.

B. McCollum. International timetabling competition 2007. `http://www.cs.qub.ac.uk/itc2007/index.htm`[Retrieved 28/11-2013], 2007b.

B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

McKinsey & Company. Analyse om øget anvendelse af it på selvejende uddannelsesinstitutioner under undervisningsministeriet - forslag til forbedringer af studieadministrative opgaver og processer. `http://www.uvm.dk/Administration/It-og-digitalisering/Ambitioes-it/~/media/UVM/Filer/Adm/PDF09/Finanslov_tilskud/AIT/090825_McK_rapport.ashx` [Retrieved 28/11-2013], June 2009. Analysis on the increased use of IT in private institutions of education under the Ministry of Education - suggestions for improvement of student administrative tasks and processes. Ordered by UNI-C and produced by the consulting firm McKinsey & Company. In Danish.

MISTA. Multidisciplinary international scheduling conference: Theory & application. `http://www.schedulingconference.org/`[Retrieved 28/11-2013], 2013.

T. Müller. Real-life examination timetabling. In *Proceedings of the tenth Multidisciplinary International Scheduling Conference (MISTA2013)*, 2013.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl. Solving the post enrolment course timetabling problem by ant colony optimization. *Annals of Operations Research*, 194(1): 325–339, 2012. ISSN 0254-5330.

B. Paechter, L. M. Gambardella, and O. Rossi-Doria. International timetabling competition 2003. http://www.idsia.ch/Files/ttcomp2002/oldindex.html[Retrieved 28/11-2013], 2002.

PATAT. International conference on the practice and theory of automated timetabling. http://www.patatconference.org/[Retrieved 28/11-2013], 2013.

N. Pillay. A survey of school timetabling research. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

G. Post, J. Kingston, A. Schaerf, L. D. Gaspero, and B. McCollum. International timetabling competition 2011. http://www.utwente.nl/ctit/hstt/itc2011/[Retrieved 28/11-2013], 2011.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194: 385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

R. Qu, E. Burke, B. McCollum, L. Merlot, and S. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1): 55–89, 2009. ISSN 1094-6136.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39 (3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

B. K. . H. Rudová. Student scheduling for bachelor state examinations. In *Multidisciplinary International Scheduling Conference VI*, 2013.

D. Ryan. It is time to enjoy the best of both worlds. In *The 46th ORSNZ Conference*, Victoria University of Wellington, New Zealand, 10-11 December 2012.

S. Schadegg. Optimized exam timetabling for danish high schools. Master's thesis, DTU Management Engineering, 2013.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

S. Suyanto. An informed genetic algorithm for university course and student timetabling problems. In *Proceedings of the 10th international conference on Artifical intelligence and soft computing: Part II*, ICAISC'10, pages 229–236, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13231-6, 978-3-642-13231-5.

Chapter 2

A Comprehensive Study of Educational Timetabling - a survey

Simon Kristiansen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, thst@dtu.dk

†MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

Abstract Educational timetabling is one of the most researched subjects within the range of timetabling problems. There has been a significant increase in efficient planning problems within educational timetabling the last couple of decades. In this paper we will highlight some of the main trends and research achievements within educational planning problems. Furthermore it is an aim to make a differentiation between the different planning problems. This survey is concentrated on the four main education planning problems; *University Course Timetabling*, *High School Timetabling*, *Examination Timetabling* and *Student Sectioning*. Firstly a presentation of educational timetabling and the main components is given. For each problem a description is given with appertaining benchmark data and recent research. The literature presented is mainly solution tested on real-life data or better yet implemented. Summarizing tables for each section are presented to give an overall view of all the literature of this survey.

2.1 Introduction

Timetabling has been a challenging and important problem within operations research for several decades and it still is. In Wren (1996) timetabling is defined as:

"*Timetabling is the allocation, subject to constraints, of given resources to objects being placed in space time, in such way as to satisfy as nearly as possible a set of desirable objectives.*"

[1] Technical report at DTU Management, report 8, 2013 (Kristiansen et al. (2013b))

Due to the broad definition the timetabling problems covers various forms of real-world problems, including *Employee Timetabling* (e.g. Balakrishnan and Wong (1990) and Meisels and Schaerf (2003)), *Rostering Problems* (e.g. Cheang et al. (2003) and Burke et al. (2004b)), *Sports Timetabling* (e.g. Easton et al. (2003) and Kendall et al. (2010)) and *Educational Timetabling* (e.g. Burke and Newall (1999), Melício et al. (2005) and Kristiansen et al. (2011)).

This survey will be concentrated on *Educational Timetabling* and the different problems within this area. Educational timetabling encompasses problems such as *University Course Timetabling*, *High School Timetabling*, *Examination Timetabling* and *Student Sectioning*, and is encountered in all institutions in the educational system throughout the world and it is one of the mostly studied timetabling problems from a practical viewpoint. The problems are very difficult and important for the institutions, and several approaches have been used to create good and feasible solutions. The purpose of this paper is to give a comprehensive overview of the different planning problems within education timetabling with the recent developments and trends. Table 2.1 lists the educational timetabling problem which we are going to discuss in this paper.

It should be mentioned that it is not attempted to perform any experimental comparison on the different methods used, only to give and overview of the methods.

Furthermore, in the effort of producing an exhaustive review of all the planning problems in the education system, we are conscious that some references have been left out and we will therefore apologize in advance. However the goal is not to list all scientific papers on this subject, but to create an overview which illustrates the different types of planning problems within the educational system and the diversity of solution methods used on these problems.

Table 2.1: Educational timetabling problems

Problem	Short description and latest survey
University Course Timetabling	Aims to break university courses into sections, and assign times, students and rooms. Either Curriculum-based or Enrollment-based Course Timetabling. McCollum (2006) motivates in bridging the gap between theory and practice within University Course Timetabling
High School Timetabling	Allocating classes to time slots, teachers and rooms to satisfy the restrictions. Pillay (2013) gives a definition and an overview of the High School Timetabling problem.
Examination Timetabling	Scheduling exams for students under the limited resources. Qu et al. (2009) is a comprehensive study of Examination Timetabling.
Student Sectioning	Assigning student to sections of courses while respecting the requests of the individual students. There is no recent survey on Student Sectioning.

2.1.1 Previous Surveys and Competitions

The literature on educational timetable includes several surveys on educational timetables. A briefly discussion of the scope of the surveys is given here sorted into chronological order:

Some of the earliest surveys include **Schmidt and Ströhlein (1980)**, **de Werra (1985)** and **Junginger (1986)**. **Schmidt and Ströhlein (1980)** provide an annotated bibliography containing more than 200 papers, and hence listing all the papers on the field up to 1979. **de Werra (1985)** introduces the various problems within education timetabling and provides with different models to the *Class-Teacher Timetabling* and *Course Timetabling* based on graph theory, while **Junginger (1986)** describes the various software products implemented for solving school

timetabling in Germany.

The first survey on *Examination Timetabling* was presented in **Carter (1986)**. None of the approaches mentioned in this survey was implemented in more than one institute. The survey was updated in **Carter and Laporte (1996)** where the approaches used on Examination Timetabling between 1986 and 1996 is summarized. The criteria for discussion of this paper, was that the solution methods should either have been implemented in a real world application or tested on real life data.

Bardadym (1996) considers computer-aided timetabling for high schools and universities from 1960 to 1995. The paper discusses the core items in the different problems and gives a small discussion on open issues for future research within timetabling.

In **Carter and Laporte (1998)** the authors has changed their focus from Examination Timetabling to Course Timetabling. The major components of the University Course Timetabling problem are described in details as well as a discussion of the primary solution methods used. The paper summarize by listing up papers which contains approaches which are either implemented or tested on real data.

In **Schaerf (1999a)** the difference between Course Timetabling and Examination Timetabling is observed as being relatively small and that the problems could be modeled using the same model. Various formulations of School Timetabling, Course Timetabling and Examination Timetabling is surveyed along with the techniques used for solving the problems.

Burke and Petrovic (2002) give an overview of some of the recent developments that have been carried out in the Automated Scheduling, Optimization and Planning Research Group (ASAP) at the University of Nottingham.

Burke et al. (2004c) gives an introduction to the field of Educational and Sport Timetabling. The paper discusses the application of graph coloring methods to the timetabling problems and thereby highlights the fact that graph coloring have been an important part of timetabling problems in several decades.

McCollum (2006) provides information regarding research on University Course Timetabling up to 2006. The aim is to motivate researchers to bridge the gap between research and practice within University Course Timetabling. The paper rounds up by listing the major challenges working within Examination Timetabling and University Course Timetabling.

Qu et al. (2009) is the far newest survey on Examination Timetabling. The basis of the paper is **Carter and Laporte (1996)** and it is therefore concentrated on papers published between 1996 and 2009. The different algorithm approaches are classified and discussed. Furthermore the paper renames the existing problem datasets to avoid a significant amount of confusion which have been a problem for many years. The paper rounds off with an estimate on future research direction within examination timetabling.

Pillay (2013) is the latest survey and the first survey concentrated only on school timetabling. The survey gives a definition of the school timetabling, the different hard and soft constraints and it gives a detailed overview on the solution methods used.

The book *Automated Scheduling and Planning From Theory to Practice* contains a chapter on Educational Timetabling (**Kingston (2013a)**). The chapter gives a brief introduction to the different educational planning problems.

Some conferences are dedicated to the art of timetabling, such as the *International Conference on the Practice and Theory of Automated Timetabling* (PATAT (2013)) and the *Multidisciplinary International Scheduling Conference: Theory & Application* (MISTA (2013)) Both conferences are held every second year.

Besides the mentioned surveys and conferences there has been three ***International Timetabling Competitions*** (ITC) on educational timetabling problems. These competitions have contributed to an increased focus on educational timetabling problems and thus the research within.

The First International Timetabling Competition was held in 2003 (ITC2003) and was regarding University Course Timetabling. The description of the problem can be found on the website of the competition (Paechter et al. (2002)). The results were presented at PATAT in 2004.

The Second International Timetabling Competition of 2007 had three tracks; Enrollment-based University Course Timetabling, Curriculum-based University Course Timetabling and Examination Timetabling (Gaspero et al. (2007), McCollum et al. (2010)) The winner of each track were announced at PATAT in 2008. Problem description can be found at the competition website (McCollum (2007)).

The third and most resent is the ITC2011 with focus on High School Timetabling (Post et al. (2012c)) and the results were presented at PATAT in 2012. The problem description and competition rules can be found at the website of the competition (Post et al. (2011)).

The results of the different competitions are described later in this paper.

From the above listing, it is seen that there exist several excellent surveys on educational timetabling. This paper concentrates on articles tested on real data or better yet implemented. I.e. it tries to follow the approach introduced in Carter and Laporte (1996) and Carter and Laporte (1998). We try to restrict the paper to only contain newly published papers, i.e. papers published since 2003, however we are aware that some of the educational problems are so sparse represented in the literature that earlier papers might be considers for these problems.

The surveys and competitions described above are listed in Table 2.1 and Table 2.2 in 2.A.

2.1.2 Outline of the paper

The educational timetabling problems are in this survey divided into four main categories: *University Course Timetabling*, *High School Timetabling*, *Examination Timetabling* and *Student Sectioning*. However, as aforementioned, other surveys have a different view and it can be discussed whether some of the categories could be combined.

The article is organized as follows. In Section 2.2 a description of the general timetabling problem in the education system is given. In the following four sections, each section treats their own planning problem. University Course Timetabling problem is described in Section 2.3 and High School Timetabling in Section 2.4. Section 2.5 elaborates on Examination Timetabling. Finally Section 2.6 describes the Student Sectioning. The conclusion of this paper follows in Section 2.7 and in 2.A all the literature for each problem are listed with appertaining comments.

2.2 Planning Problems and the Components

The definition *timetabling* is often used when solving problems with some form of personnel allocation, such as University Course Timetabling. Due to the personnel allocations, many people are affected by the timetables created. For the combined planning problems in the educational system four main stakeholders are identified. Each with their own set of aims and wants.

- *The administration*: Solves the planning problems and sets the minimum standards that the timetables must conform to. The capacity of the room, consecutive exams, etc.

- *The students*: This is by far the largest in volume of the four stakeholders. The students seldom have much influence on the outcome of the different planning problems. And due to the number of students involved it is difficult to declare what is the best timetable for the students as they all might have different wants and needs.

- *The teachers*: The teachers have a little more influence on the outcome than the students and they do also requests a compressed timetables. Furthermore they might have some restrictions on which day and time they want to give lessons or exams.

- *The departments/sections*: Often only relevant for universities as the high schools rarely are organized with departments or sections. The department can have some restrictions or requests for the exam and course timetabling.

For all the various planning problems some restrictions always apply, these are the so-called *"first-order conflicts"*. E.g. no person can be at more than one place at any time. These first-order constrains are always present in some form. Besides these constraints, there is a great variation between the problems, but also within a given problem there is great variation due to different universities, high schools and/or the Ministries of Education of the respective countries. Hence the other constraints within educational timetabling are many and varied. In the following some of the most common types are listed.

- **Resource assignment:** Resources are rarely preassigned to lectures and in these cases the assigning of resources is a part of the decision problem. The most common resources to assign are rooms. Some lectures prefer or require to be held in a particular class. Furthermore the number of students in a given room must not exceed the capacity of the room.
- **Continuity:** Constraints that ensure that certain features of the timetables are constant or predictable. E.g. lectures of the same course should be scheduled in the same room or at the same time of days.
- **Compactness:** These constraints are designed to produce a more compact timetable for both students and teachers. For the students a compact timetable is one with a low number of idle time slots in between lectures. A teacher might prefer to have all his lectures in fewer days and thereby have some days of.
- **Spreading:** Often used as contrary to compactness. Meetings should be spread out in time. However it is also used on single meeting. If the same meeting should be repeated twice or more times in the timetable it is often preferable if these similar meetings a spread out.
- **Time assignment:** These constraints are used for assigning meeting to a time. This can be used to specify days on which teachers are unavailable or if one particular meeting must take place after another one, e.g. exercises should be placed right after the corresponding lectures. It is also used for meetings that should be held simultaneous.

These types of constraints are somewhat presented in educational timetabling as we will see in the following sections.

2.2.1 Annual Cycle of Educational Timetabling Problems

Whether it applies to high schools or universities, all the planning problems at an educational institute are recurring annual administrative tasks. There are differences in the planning problems that the various institutions have in their annual cycle, but in general the annual cycle of educational planning problems contains three main subjects; *Student Sectioning* followed by *University Course Timetabling* or *High School Timetabling* and finally the *Examination Timetabling*. Figure 2.1 illustrates the annual cycle of planning problems of a typical educational institute.

For some institutes some of the planning problems might be combined or swap positions in the cycle, or additional planning problems might be added to the cycle. At some universities Student Sectioning is used before and after the course timetabling.

2.3 University Course Timetabling

University Course Timetabling problem is the task of assigning lectures of courses to time slots, rooms and other resources, subject to constraints applied for the single student. The large size and structure of universities often makes the problem of solving the course timetabling so complicated that it is solved by several experts scattered across different faculties and departments. Due to the complexity of the problem, it is quite common that the universities re-use the solution from previous years when solving the problem.

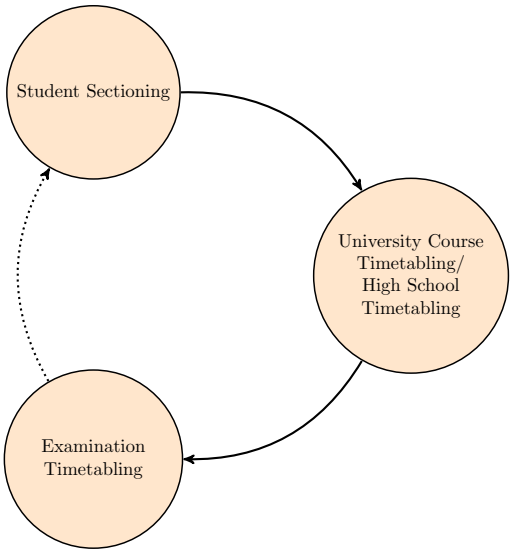


Figure 2.1: A typical annual cycle of planning problems at an educational institute. Each year starting with Student Sectioning, then Timetabling (either University Course Timetabling or High School Timetabling) and finally the Examination Timetabling before repeating for the next school year.

The University Course Timetabling is often divided into two general approaches; the *Curriculum-based University Course Timetabling* and the *Enrollment-based University Course Timetabling*.

Courses which can be taken in combination, because they are needed to satisfy the degree rules of the given study, are called curricula. And assigning courses to time slots based on this is called Curriculum-based Course Timetabling. The problem consists of creating the weekly schedules of lectures where conflicts between the courses are set accordingly to the curriculum specified by the university. Gaspero et al. (2007) gives a short description of the Curriculum-based Course Timetabling Problem applied for Track 2 of ITC2007. Several research papers have been working on this university course approach (Eg. Burke et al. (2012); Cacchiani et al. (2013); Lü et al. (2011); Lach and Lübbecke (2012)).

The second approach is the Post Enrollment-based Course Timetabling. Based on the enrollment data of each individual student, it is determined where courses are placed in the timetable such that all students can attend the events on which they are enrolled. This approach has been given various names in the literature including the Class Timetabling Problem and Event Timetabling. Lewis et al. (2007) gives a description of the Post Enrollment-based Course Timetabling Problem applied for Track 3 of ITC2007. Papers working on the Post Enrollment-based Course Timetabling include Cambazard et al. (2008); Ceschia et al. (2012) and Nothegger et al. (2012).

The two approached are not necessary mutually exclusive. The universities can use the curriculum based approach in an early state to get the basic structure of the timetable, and then later on use the enrollment-based approach to improve the timetable.

Though the literature often is divided into the two approaches, many of the constraints remain the same. Below some of the most common constraints for University Course Timetabling are listed.

- Primary hard constraints
  - No 1st-order conflicts for students and teachers.
  - Only one event is put into each room in any time slot.
  - Rooms should satisfy all of the features required by the event.
- Primary soft constraints

- The capacity of the room should be respected.
- Compact timetable of each student or curricula.
- Room stability. Lectures of same course should be assigned the same room.
- Time and/or room preferences.

2.3.1 International Timetabling Competition

In the First International Timetabling Competition in 2003 (ITC2003), the focus was Post Enrollment-based University Course Timetabling. The winner became Kostuch (2004) with a Simulated Annealing based heuristic.

The most recent competition on University Course Timetabling is the Second International Timetabling Competition in 2007 (ITC2007) (McCollum et al. (2010)). The competition consisted of three tracks where two were upon University Course Timetabling, with enrollment-based in Track 2 and curriculum-based in Track 3. Track 1 treated the Examination Timetabling problem.

The winner of Track 2 was a team from Ireland solving the Enrollment-based University Course Timetabling using a hybridized local search algorithm with a Constraint Programming approach in a Large Neighborhood Search scheme to address the hardness of finding feasible solutions (Cambazard et al. (2008)).

Track 3 was won by Tomáš Müller using a hybrid heuristic using three phases (Müller (2009)). Firstly a feasible initial solution is found using an Iterative Forward Search algorithm. In phase two the local optimum is found using a Hill Climbing algorithm and third phase is using Great Deluge techniques. This heuristic solver was among the finalist for all three tracks, and won Track 1 and 3.

2.3.2 Benchmark Data

There is a major lack of benchmark data for University Course Timetabling. ITC2007 contains datasets for both the Enrollment-based Course Timetabling and the Curriculum-based Course Timetabling. The datasets are available at the website of the competition McCollum (2007).

- **ITC2007 Track2 - Enrollment-based Course Timetabling**
  Twenty-four datasets are available; all were created using an automated problem generator designed by the competition organizers, and all are known to feature at least one perfect solution - that is, a solution with no hard or soft constraint violations. The drawback of this benchmark is that it is generated data and not real-life.

- **ITC2007 Track3 - Curriculum-based Course Timetabling**
  Twenty-one instances were released for this track. All instances are real data and come from the University of Udine and for all instances there exists at least one feasible solution.

Bonutti et al. (2012) collects the twenty-one instances of ITC2007 and four instances from Gaspero and Schaerf (2003) together with seven new instances mainly from Italian universities. The datasets are all real cases from various universities, mainly from University of Udine. In the following these datasets are denoted as one benchmark called **The Udine benchmark data** UdineBenchmark (2013).

- **The Purdue benchmark data**
  On the website of the educational scheduling system UniTime, there is access to real data from the University of Purdue, United States. The benchmark consists of data sets for each department and one for the combined problem of three years (PurdueBenchmark (2013)).

2.3.3 Recent Research

University Course Timetabling is one of the most researched subjects within educational timetabling. The last decade many of the used methodologies represent some sort of hybridization of a number of techniques. In the following the scientific papers on University Course Timetabling of the last decade are categorized into the main methodologies used.

Swarm Intelligence Algorithms

Swarm intelligence algorithms belong to the family of population based techniques. Based on some sort of agents interacting locally with one another and with their environment. E.g. in Ant Colony algorithms, the ants search the shortest path to food by laying pheromones on the way. The shortest path is the path with the strongest level of pheromones.

In Socha et al. (2003) two different Ant Colony methods is used to solve the University Course Timetabling; Ant Colony System and MAX-MIN Ant System. In each step of both algorithms, every ant constructs a complete assignment of events to time slots using heuristics and pheromone information. The timetables are then improved using a local search procedure.

Ant Colony is also the method used in Nothegger et al. (2012) to solve the Post Enrollment-based Course Timetabling problem. The algorithm uses two distinct but relatively compact pheromone matrices in combination with an effective procedure to exploit their information in the heuristic solution construction. The algorithm was tested on the benchmark datasets from ITC2007 with the same terms as the competition. It had the best solution for 11 out of the 24 instances compared to the 5 finalists, including ties, which would have given a fourth place. However the algorithm did show a large variation in the solution quality as it produced both the best and worst solution for several instances.

In Turabieh et al. (2010) a simulation of a fish swarm is applied to the same University Course Timetabling definition used in Socha et al. (2003). The idea of Fish Swarm Intelligent is to simulate the behavior of fishes while searching for food. The movements of the fishes are based on a Nelder-Mead simplex algorithm. Two types of local search were applied to enhance the quality of the solution. A multi decay rate Great Deluge and a steepest descent algorithm. The same method was later used to solve the Examination Timetabling problem (Turabieh and Abdullah (2011b)).

Another Swarm Intelligence is the Particle Swarm optimization, which is used on University Course Timetabling in Shiau (2011) and in Chen and Shih (2013). The Particle Swarm Intelligence algorithm consists of a swarm of particles in the space. The position of a particle is indicated by a vector which presents a solution and the movements of the particles are guided by their own best known position in the search-space as well as the entire swarm's best known position. Shiau (2011) solves the course timetabling problems for a typical university in Taiwan by applying a hybrid particle swarm optimization. Each particle is updated on the basis of continuous particle swarm optimization formulas and local search. Chen and Shih (2013) evaluates on two different kind of particle swarm heuristics used on the problem, an inertia weight version and a constriction version. An interchange heuristic was applied to enhance the quality of the solution. The algorithms were tested on a single dataset where the Constriction Particle Swarm with interchange heuristic performed best.

Evolutionary Algorithms

Evolutionary algorithms do also belong to the family of population based techniques. The most common evolutionary algorithm is the Genetic Algorithm. Genetic algorithms are search heuristic that mimics the process of natural selection.

In Nurmi and Kyngas (2008) the Curriculum-based Course Timetabling problem is solved using an evolutionary algorithm. Firstly the Curriculum-based Course Timetabling is turned into the School Timetabling problem using a conversion scheme. The problem is then solved using a genetic algorithm (Nurmi and Kyngas (2007)). The algorithm consists of one mutation operator and no

recombination operators. The most important features of the algorithm are a greedy hill-climbing mutation operator and a adaptive genetic penalty method.

Suyanto (2010) describes an informed generic algorithm for the University Course Timetabling and Student Sectioning. Firstly a greedy heuristic creates some feasible solution were all the hard constraints are satisfied. Then a directed mutation scheme is used to reduce the violations of the soft constraints while keeping the solution feasible.

Local Search Algorithms

Local search based techniques are classified as meta-heuristics and the algorithms move from solution to solution in the search space by applying local changes. Local search algorithms include methods such as Tabu Search, Simulated Annealing (SA) and Greedy Randomized Adaptive Search Procedures (GRASP). Within educational timetabling local search has been widely used and in the following some of the local search based techniques used on University Course Timetabling in the last decade are presented.

Gaspero and Schaerf (2003) propose a set of multi-neighborhood search strategies to improve local search capabilities. It is showed how different neighborhoods operators can be combined. The operators chosen are Tabu Search and Hill-Climber. The combinations used are neighborhood-union, neighborhood-composition and token-ring search. A perturbation operator called "kicker" is used to help improve the search and hence avoid local optima. The algorithm is tested on four real instances from the School of Engineering at Udine University.

Another neighborhood analysis is made in Lü et al. (2011). Four neighborhoods are investigated based on three evaluation criteria: percentage of improving neighbors, improvement strength and search steps. The neighborhoods in this paper is SimpleMove, SimpleSwap and two kinds of KempeMove. To understand the behavior of the neighborhoods and the combinations, they are tested using a steepest descent algorithm. To further evaluate the impact, a series of experiments are conducted using three algorithms: Tabu Search, Iterate Local Search and Adaptive Tabu Search.

Simulated Annealing is another local search algorithm. The idea is to search a wider area of the search space by allowing the algorithm to accept worse moves with a higher probability in the beginning of the search. The acceptance criterion is controlled with a temperature which decades based on a cooling schedule. Kostuch (2004) participated in ITC2003 and won using a heuristic based on Simulated Annealing. Firstly a feasible timetable is constructed, and the timetable is then improved using SA.

Murray et al. (2007) transforms the University Course Timetabling problem at the Purdue University into a constraint satisfaction and optimization problem (CSOP). CSOP consists of a set of variables having finite domain, a set of hard constraints and an objective function. The solver used an iterative forward search algorithm. The paper is a part of the research embedded in the system UniTime.

In Ceschia et al. (2012) the Post Enrollment-based Course Timetabling problem is solved using SA. The SA used is with probabilistic acceptance and a geometric cooling scheme. The algorithm is tested on the benchmark of ITC2003, ITC2007 and the data used in Lewis and Paechter (2005).

Variable Neighborhood Search (VNS) algorithms are used to solve the Curriculum-Based Course Timetabling in Nguyen et al. (2011). Eight Variable Neighborhood Search algorithms with different implementing strategies were created. A basic VNS and seven popular variants, where the Fleszar-Hindi extension of the basic VNS in general was the most effective in solving the 14 instances from University of Science in Vietnam.

Chiarandini et al. (2006) uses a hyper-heuristic which combines various construction heuristics, Tabu Search, Variable Neighborhood Descent and Simulated Annealing. The algorithm is divided in two parts. In the first part construction heuristics are used to solve the hard constraints of the problem. In the second phase the penalty of soft constraints are improved by first using Variable Neighborhood Search, and then using Simulated Annealing. The authors make use of

the racing procedure, the *F-race* method, to automatically tune the parameters of the algorithm. The algorithm is tested on the benchmark of ITC2003.

Graph Coloring Algorithms

In Burke et al. (2007) a graph-based hyper heuristic is developed for University Course Timetabling and Examination Timetabling. A Tabu Search approach is employed to search for permutations of graph heuristics which are used for constructing timetables. For the University Course Timetabling the approach is tested on the datasets from ITC2007 and the results were competitive with the previous approaches reported in the literature.

Many papers work with a two-phase approach for solving University Course Timetabling. A construction phase and an improvement phase. Where most papers are concentrated on the improvement phase, Azlan and Hussin (2013) focus on creating as good initial solutions as possible to solve the Curriculum-based Course Timetable. Two different construction heuristics based on Graph Coloring are created; largest degree and largest weighted degree. The two heuristics are tested on the datasets from ITC2007 and both are able to find feasible solutions for most of the cases. Largest weighted degree being the one performing best.

Exact Methods

The drawback of using some sort of heuristic to solve timetabling problems is the lack of capability to issue certificates of optimality or the quality of the solutions found. This is made possible using *Integer Programming* methods such as decomposition techniques.

The University Course Timetabling at Ohio University's College of Business is solved in Martin (2004) using Integer Programming (IP). The problem is solved using CPLEX.

In Daskalaki et al. (2004) an Integer Programming formulation for the University Course Timetabling is developed. The formulation is based on the timetabling problem in Engineering Schools of Greek universities. It was possible to solve the single real case of the problem with success using CPLEX. Daskalaki and Birbas (2005) use a two stage relaxation procedure to solve the problem. In first stage the relaxation is performed and it concerns the constraints that ensure consecutiveness for the multi-period sessions assigned to a given course. These constraints are recovered during second stage where a subproblem for each day is solved for local optima.

Avella and Vasil'Ev (2005) describes a Branch-and-Cut algorithm for the University Course Timetabling problem and two cutting planes are derived, Clique inequalities and Lifted Odd-Hole inequalities, to tighten the initial formulation.

Qualizza and Serafini (2005) propose an Integer Programming approach based on Column Generation. Each column represents a weekly timetable of a single course. The master problem then contains all the constraints referring to classroom occupancy and non-overlapping in time of courses. Each subproblem contains the constraints related to a single course timetable and hence creates weekly timetable for a single course. A Branch-and-Bound method is used to ensure feasibility of the solution.

Al-Yakoob and Sherali (2007) takes its origin at Kuwait University. The paper formulates a Mixed Integer Programming (MIP) model for the University Course Timetabling problem which is able to design class timetables that have a number of desirables features related to minimizing class conflicts, providing good patterns, and enhancing traffic flow. Moreover, gender policies are considered when creating sections of courses.

Schimmelpfeng and Helber (2007) solves the University Course Timetabling problem at the School of Economics and Management at Hannover University by modeling the problem as an IP model and solved using a state-of-art MIP solver. An anonymous satisfactory survey among the faculty was initiated and in general the response was positive.

In Lach and Lübbecke (2012) the Udine benchmark datasets on Curriculum-based Course Timetabling are solved using a two stage decomposition approach. The theoretical background of the decomposition is given in Lach and Lübbecke (2008). The outline of the two stage decomposition

is firstly to assign lectures to time slots (Stage1) and then assign rooms to the lectures (Stage 2). The approach has proven to be an effective method for solving the Curriculum-based Course Timetabling.

Hao and Benlic (2011) generates lower bound for the ITC2007 benchmark data on the Curriculum-based Course Timetabling problem. They present a new partition-based approach based on the divide and conquer principle consisting of three phases. Firstly, courses are partitioned into a fixed number of subproblems using an iterative Tabu Search. In second stage the subproblems are formulated as integer linear problems using the formulation of Lach and Lübbecke (2012). And finally the subproblems are solved using an IP solver. Better lower bounds were found for all instances except two. The found lower bounds can be used to estimate the quality of the solutions obtained with some of the various heuristic approaches aforementioned.

In Burke et al. (2012) a Branch-and-Cut procedure is suggested for the Udine benchmark datasets. The Integer Programming formulation is the same as for Burke et al. (2010b). The procedure reduces the number of variables necessary to formulate the soft constraints. Using the Branch-and-Cut it is possible to achieve good lower bounds must faster than solving the initial IP model.

Cacchiani et al. (2013) is the latest research paper working on improvements of lower bounds on the University Course Timetabling problem. The bounds are obtained by splitting the objective function into two parts and formulate an integer linear programming models for both. The solution of each is obtained by using a Column Generation procedure. The global bound is then obtained by summing up the corresponding optimal values. By comparing the results with results from previous research on the instances used at ITC2003 and ITC2007, it is proven that this method is able to improve some best-known lower bounds and that for some instances the best known solutions is indeed optimal (or close to).

Software systems

Only few systems which solve the University Course Timetabling have been published in detail.

Carter (2001) solves the Enrollment-based Course Timetabling for the University of Waterloo, Canada, and has been used since 1985. The algorithm behind is a three-phase method. First, the Student Sectioning problem is solved aiming at minimizing the number of pairs of sections with students in common. Secondly, the sections are assigned time slots and finally the timetabling is improved for each student using single student timetabling, where the timetable of each student is improved individually.

Dimopoulou and Miliotis (2001) has created a system to solve the University Course Timetabling and the Examination Timetabling at Athens University of Economics and Business. The course timetabling problem is modeled and solved using a Branch-and-Bound based computer code. In most of the tests performed the optimal solution was found in less than one minute.

Another system is UniTime. UniTime is an open-source comprehensive educational scheduling system for universities. It offers both Curriculum-based and Enrollment-based Course Timetabling, as well as Examination Timetabling and Student Sectioning, and is used at several universities. The website of UniTime features all the papers published for the system (UniTime (2013)).

Rudova and Murray (2003) is some of the earliest work of UniTime. Of later research Rudová et al. (2011) solves the Post Enrollment-based Course Timetabling using the Iterative Forward Search (IFS) algorithm described in Müller et al. (2005). In Müller and Rudová (2012) the Curriculum-based Course Timetable is solved by transforming its model into the enrollment model and using a local search algorithm for generating the corresponding enrollments. The algorithms of both papers are implemented in UniTime.

All the references mentioned in this section are listed in Table 2.3 in 2.A.

2.4 High School Timetabling

High School Timetabling is the problem of allocating classes to time slots, teachers and rooms to satisfy the hard and soft constraints. At the high schools the students are grouped into classes prior to the timetabling problem. The students of the classes are usually occupied together for all the lectures of their given class.

The most important components of the High School Timetabling problem are quite similar of those of University Course Timetabling. However there are some significant differences. Firstly, Student Sectioning is considered as part of the University Course Timetabling, whereas at the high schools it is considered as a separate problem. Secondly we have the grouping of students. High schools are timetabling classes, whereas at universities the students are often timetabled individually and at high schools clashes are not acceptable for the classes, whereas it can be acceptable if some students have clashes at universities. Another difference is the use of teachers. At the high schools the teachers are teaching full time, whereas the universities the teaching is often a small part of the professors/lectures workload.

Some of primary hard and soft constraints of High School Timetabling are listed below:

- Primary hard constraints
  - No 1-order conflicts. A student cannot attend two courses that are overlapping in time.
  - Classes/events must be scheduled for the required number of times for each subject.
  - Classes/events must be assigned a resource/room.
  - Room capacity. Classes can only be assigned rooms of which the capacity suits the class size
- Primary soft constraints
  - Limit idle periods for student and/or teachers.
  - Lessons spreading.
  - Resource/times preferences

The XHSTT format, which is described in the next section, operates with 16 different constraints, which can either be denoted hard or soft.

2.4.1 International Timetabling Competition

As mentioned the Third International Timetabling Competition (ITC2011) had the High School Timetabling problem as topic. The concept of the idea was to increase the focus on High School Timetabling and the XHSTT format. The competition was launched in October 2011 and features three rounds.

1. In the first round of the competition the competitors should construct all-time-best solutions to the instances of the XHSTT-2012 archive.
2. The second round of the competition will work on the instances of the XHSTT-ITC2011 archive and 15 hidden instances. The participant could use any programming language and free third party software (excludes CPLEX, Gurobi etc.). A time limit of 100 seconds is given.
3. Third round consists of constructing all-time-best solutions to the hidden instances.

Deadline for submission of the contribution was May 2012 and the results was announces at PATAT in August 2012.

2.4.2 Benchmark Data

Access to public datasets of High School Timetabling has been very limited. Previously research was done for new unique problems each time or on artificial instances. The first accessible real life instances are the Beligiannis data sets.

- **Beligiannis data sets - Greek high schools**
  The archive consists of 7 datasets from Greek high schools and is used in Beligiannis et al. (2008). Each datasets contains a requirements matrix specifying the number of times each teacher must teach each class.

Due to the lack of exchangeable benchmarks in a uniform format, a group of researchers agreed on developing an XML-standard for the High School Timetabling problem, this resulted in the *XHSTT* format.

- **XHSTT**
  The project on creating XHSTT is described in the paper of Post et al. (2012a), and the format was as mentioned used for ITC2011. The objective is to minimize the number of violation of hard and soft constraints. The description of the format of XHSTT is available at the homepage (Post (2013)) along with nearly 40 instances. An evaluator for instances is available at at the web-page of Jeffrey Kingston (Kingston (2013b)).

2.4.3 Recent Research

Research articles on High School Timetabling are mostly limited to a single high school or a single country. In this section we will look at some of the research papers which have been conducted. For the High School Timetabling, the papers are categorized in order of the origin of the problems. This does not necessary means that the problems at the different countries are not the same, but that the solution methods only have been tested on instances of the given country.

Pillay (2013) is another comprehensive survey which is advised for additional details on publications on High School Timetabling. In Pillay (2013) the literature is categorized by the used methodologies.

Country Specific Research

Australia
The Australian case is solved in Boland et al. (2008) by creating two integer linear programming models to solve the course blocking and population problem.

Kingston (2005) solves seven instances of Australian high schools using a tilling algorithm with hill-climbing. The problem is of allocating meetings (teachers and classes) to times. Resources are added to the meeting using an alternating path algorithm. The XHSTT archive consists of three instances from Australia and they are the only instances where teachers workload are constrained.

Brazil
The Brazilian case of School Timetabling is one of the most represented cases in the literature. Filho and Lorena (2001) use a constructive genetic algorithm to solve two Brazilian high schools, whereas Souza et al. (2003) use a greedy algorithm to find a good initial solution and then a Tabu Search for improving this solution.

Santos et al. (2004, 2005) creates a Tabu Search algorithm with two different memory based diversification strategies. The papers show that the diversification strategies improve the robustness of the Tabu Search.

In Bello et al. (2008) the High School Timetabling problem is treated as a Graph Coloring problem and solved using a Modified Tabucol, where Tabucol is a Tabu Search for Graph Vertex Coloring. The method was tested on five Brazilian high schools. Moura and Scaraficci (2010) solves the High School Timetabling for three Brazilian high schools using a basic GRASP heuristic, followed by a path-relinking improvement. Santos et al. (2012) present Column Generation as an

approach for establishing bounds for a set of datasets originating from Brazil. It was a Brazilian team that won the ITC2011, Fonseca et al. (2012).

Currently, the XHSTT archive consists of seven datasets from Brazil.

Denmark

Denmark is one of the new entrants within High School Timetabling. Sørensen and Stidsen (2013) is the first paper working on Danish high school and it describes a complex MIP model of the problem and establishes computational results for 100 real-life instances using Adaptive Large Neighborhood Search. The solution approach is implemented in cloud-based administrative software and is available for the majority of all Danish high schools.

Sørensen and Dahms (2014) suggest a two-phase decomposition (created by Lach and Lübbecke (2012)) for solving the high schools in Denmark. In the first stage lectures are assigned to time slots and in the second stage rooms are assigned to the lectures. The results show that the approach is more effective than solving the original IP model in terms of both the obtained solutions and the obtained lower bounds.

At present four instances of Danish high schools are available in the XHSTT archive.

Germany

In Bufé et al. (2001) the German high schools are treated with an evolutionary heuristic. Firstly an evolutionary algorithm searches the space of all permutations of the events from which the high school timetable is created. The solution is then improved using local search with specific mutation operators.

Lohnertz (2002) uses a hybrid approach with a combination of Tabu Search and Graph Vertex Coloring. Whereas Wilke et al. (2002) uses a genetic algorithm for solving the German high school timetabling.

A Tabu Search algorithm for the German timetabling problem is presented in Jacobsen et al. (2006). The initial solution is created using a construction heuristic with a graph coloring algorithm. The solution is then improved with the Tabu Search algorithm.

So far as we know none of the German instances are available in an online archive.

Greece

The Greek high schools are well researched and the XHSTT archive is presented with seven instances from Greece.

In Valouxis and Housos (2003) constraint programming are used in combination with local search for Greek high schools. Papoutsis et al. (2003) creates an Integer Programming model of the problem and solves it using Column Generation.

In Beligiannis et al. (2008) an evolutionary algorithm has been created for the High School Timetabling problem in Greece and this system has been extended with a user-friendly interface (Moschopoulos et al. (2009)). In later work Beligiannis et al. (2009) implement a genetic algorithm to solve the same case. Both algorithms is inherent adaptive, which means that the user is able to assign weights to the different constraints. Both algorithms have been tested on seven real timetables used at schools in the city of Patras and has been made available online (See Beligiannis benchmark, Section 2.4.2).

In Raghavjee and Pillay (2009) and Raghavjee and Pillay (2010) a generic algorithm is used to solve the high school timetabling problem in Greece and South Africa, respectively. The Greek datasets are from the Beligiannis benchmark datasets.

Birbas et al. (2009) uses a hybrid approach to solve the problem at a secondary Hellenic school. The first phase solves the Shift Assignment Problem where teachers are allocated to shifts, and the second phase solves the High School Timetabling. Both phases are solved using Integer Programming.

Zhang et al. (2010) uses a simulated based algorithm for the High School Timetabling problem. The algorithm is tested on two sets of benchmark instances. One randomly created and one real-life from Greece.

Valouxis et al. (2012) describe a two-phase approach based on MIP used to solve the Greek case of the High School Timetabling problem. This includes two instances which are part of the XHSTT project, which were both solved to optimality (solutions were found with an objective value of 0).

Greece is represented in the XHSTT archive with seven instances.

Italy

Colorni et al. (1998) compares different solution methods for solving two Italian instances of the High School Timetabling problem. The different methods tested are various versions of Simulated Annealing and Tabu Search, and genetic algorithms with and without local search. The genetic algorithm with local search and the Tabu Search based on temporary problem relaxations outperforms the other Simulated Annealing approach and handmade solutions. The genetic algorithm with Tabu Search being the best.

Schaerf (1999b) has implemented a hybrid approach. After the initial timetable is created a randomized non-ascendant search is applied to improve the solution. When no better solution can be found, a Tabu search is applied. The two methods are repeated sequentially until there are no further improvements of the timetable. The solution method is tested on one artificial school and two Italian high schools.

Avella et al. (2007) uses a hybrid algorithm consisting of a Simulated Annealing for the initial solution and Very Large Neighborhood Search for improvements. It is tested on two real life instances from Italy.

Italy is represented with one instance in the XHSTT archive.

Netherlands

Post and Ruizenaar (2004) use a combination of clustering and Branch-and-Bound algorithms to solve the school timetabling problem for a Netherlands secondary school. The first step is to construct clusterschemes. A clusterscheme contains clusterlines with optional subjects that can be taught in parallel. The second step is the Branch-and-Bound algorithm which constructs the school timetable.

In de Haan et al. (2007) a three-phase approach is use to generate a timetable for a Netherlands high school. The first phase constructs cluster schemes for optional subjects, in the second phase a feasible schedule is constructing by assigning all lessons to time slots, such that there are no clashes. Finally the third part makes improvements of the timetable.

Post et al. (2012b) presented a cyclic transfer algorithm for the High School Timetabling problem. The methods are tested on four Dutch high schools and one English high school.

In the XHSTT archive the Netherlands are represented with four instances.

Portugal

In Fernandes et al. (1999) the Portuguese High School Timetabling problem is solved using evolutionary algorithms. The genes which cause hard constraint violations are denoted *bad genes* and by introducing bad genes mutation it is possible to improve the algorithm in both speed and solution. Melício et al. (2006) developed the software tool THOR. THOR consists of a graphical user interface, an automatic scheduler and a relational database. The automatic scheduler is using a greedy algorithm to establish an initial solution and then improving that using Simulated Annealing. The system is used by more than 100 Portuguese high schools.

Other country specific methods

Besides the mentioned countries, High School Timetabling has been briefly touched in other countries as well.

Wood and Whitaker (1998) solves the timetabling problem for secondary schools in New Zealand where student requests are an important part of the problem formulation. The problem is formu-

lated as a non-linear goal program and solved in several stages using different heuristics such as Simulated Annealing and the Hungarian assignment algorithm.

Yigit (2007) uses a hybrid genetic algorithm for solving the High School Timetabling problem for the Technical and Vocational High Schools in Turkey.

In Ribic and Konjicija (2010) a two-phase approach to modeling the timetable problem is presented. In the first phase, the classes are allocated to days, and during the second phase, each class in a day are allocated into time slots. The approach is tested on a test case from a Croatian secondary school.

Nurmi and Kyngas (2007) use an extension of a hybrid hill-climbing genetic algorithm. The extension of the algorithm is a Simulated Annealing for choosing periods "intelligently". The algorithm is tested on data from Finnish high schools. Finland is represented in the XHSTT archive with six instances.

The timetabling problem for the Vietnamese high schools were solved in Minh et al. (2010) using a greedy algorithm for the initial solution and then improve the solution using Tabu Search.

The XHSTT-format

Most of the studies on High School Timetabling have resulted in successful solvers, however a big drawback is that many of the papers have only been applied to one problem or one country. One of the reasons for this is the lacking of benchmark data. Another reason is the desire of the single school to get its own problem solved and hence the papers are not interested in a more general approach. By the XHSTT format with corresponding benchmark datasets it is now possible to model a single problem in a standard format and to tests the solution approach on several cases.

As mentioned ITC2011 considered the High School Timetabling problem based on instances of the XHSTT format (Post et al. (2012c)). Four teams made it to the final round: The overall winner (Team Goal) used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al. (2012)). Participant from the University of Nottingham (HySTT) used a method based on Hyper-heuristics (Kheiri et al. (2012)). Team Lectio from Denmark used Adaptive Large Neighborhood Search (ALNS) (Sørensen et al. (2012)) and Romrös and Homberger (2012) (Team HFT) from Germany used an Evolutionary Algorithm. The specification of ITC 2011 and the results are described in Post et al. (2013)

Pimmer and Raidl (2013) describe a 'timeslot-filling' heuristic for XHSTT, which iteratively fills selected timeslots with sets of events. Two state-of-the-art solutions were found for instances of the archive XHSTT-2012.

Fonseca et al. (2013) have made some improvements on their work from ITC2011 which exceeds their previous work (Fonseca et al. (2012)). The new algorithm is a stagnation free Late-Acceptance Hill Climping algorithm. By combining the new approach with the Simulated Annealing from Fonseca et al. (2012) it provides the best results for some XHSTT datasets of ITC2011.

Kristiansen et al. (2013c) is the first paper working on an exact method for solving problems of the XHSTT format. The paper shows that the complex XHSTT format can be formulated as a Mixed Integer Programming model and solved using the state-of-the-art MIP solver, Gurobi. It was possible to find two new optimal solutions and prove optimality of four previously known solutions. Furthermore, lower bounds were established for 11 datasets.

All the references mentioned in this section are listed in Table 2.4 and in Table 2.5 in 2.A.

2.5 Examination Timetabling

Examination Timetabling is the problem of scheduling a given number of exams to a limited number of time slots. Each course has one event representing the exam. The main problem is to avoid clashes in each student's examination timetable and to make sure that they have sufficient preparation time for each exam.

It is mainly the university course timetabling which are discussed in the literature and in Schaerf (1999a) the difference between Examination Timetabling and University Course Timetabling is observed to be relatively small, as both is of assigning event/exams to time slots and resources. However, it is broadly accepted to distinguish between the two due to the characteristics of the examination timetabling. University Course Timetabling pursuit a compact timetable whereas Examination Timetabling pursue more spreading between events for each student. The time between two exams is the preparation time the student has for the next exam. Furthermore, there is only one exam per course and there can be more than one exam in a single room. And a student can only attend one exam a day.

Below is listed some of the most common constraints used in Examination Timetabling.

- Primary hard constraints
  - 1-order conflicts cannot be accepted in Examination Timetabling.
  - Resources needs to be sufficient for the examinations (e.g. room capacity, enough rooms).
- Primary soft constraints
  - Spreading versus compact.
  - Time requirements. Exams can/cannot be in certain time slot.
  - Consecutive exams.
  - Resource requirements
  - Limited number of students and/or exams in any time slot
  - Ordering of exams must be satisfied
  - Only exams with same length can be located in the same room in the same time slot
  - Exams required taking place at the same time, on the same day or at the same location
  - As early/late as possible
  - Splitting the exams over similar locations

2.5.1 International Timetabling Competition

Aforementioned, the International Timetabling Competition in 2007 consisted, as mentioned, of three tracks, where the first track was regarding Examination Timetabling (McCollum et al. (2010)).

Eight datasets were given to the competitors, four were given immediately and four were given two weeks before competition deadline. The solution methods submitted were then tested on four hidden datasets. As mentioned in Section 2.3 it was the contribution of Müller (2009) that won Track 1 and Track 3 of the competition. The algorithm was a hybrid heuristic consisting of three parts, an Iterative Forward Search, a Hill Climbing and the Great Deluge techniques.

The datasets were provided by the EventMap research group and eight of the datasets are made available as benchmark, at the ITC2007 website (McCollum (2007)).

2.5.2 Benchmark Data

As Examination Timetabling has a high research interest it had let to some establishment of a variety of different benchmark problems. These benchmarks have made it possible to create scientific comparison between different approaches to the Examination Timetabling and thereby exchange of research achievements. The goal of this section is only to give a brief description of the known benchmark of Examination Timetabling. This paper will use name of the benchmarks as they were renamed in Qu et al. (2009) to prevent further confusion between papers. The majority of the benchmarks are available online (ExamBenchmarks (2013)).

- **The Toronto benchmark data**
  Carter et al. (1996) introduced a set of 13 real-world Examination Timetabling problem (3 from high schools and 10 from universities).

  The Toronto benchmark data have two variants of objectives: 1: Minimize the number of used time slots needed for the problem. 2: Minimize the average cost per student.

- **The Nottingham benchmark data**
  In Burke et al. (1996) a modification of the objective on six of the data sets from Carter et al. (1996) were introduced as benchmark along with the Examination Timetabling data from 1994 at the University of Nottingham.
  The objective is to minimize the number of students having two consecutive exams.

- **The Melbourne benchmark data**
  Merlot et al. (2003) introduced two new datasets from the University of Melbourne at the fourth conference of Practice and Theory of Automated Timetabling (PATAT2003). The benchmark consists of two datasets which have two time slots for each of the five workdays.

  The objective is to minimize adjacent exams on the same day or overnight.

- **The Purdue benchmark data**
  The newest benchmark within Examination Timetabling is given by Müller (2013). Nine datasets from Purdue University are introduced, and each dataset have 29 examination periods, all 2 hours long. PurdueExamBenchmark (2013)

2.5.3 Recent Research

Within educational timetabling, Examination Timetabling is a much researched subject. Qu et al. (2009) is an excellent survey of examination papers from 1996 to 2009 and we recommend the reader to read this paper for additional details on the research of Examination Timetabling.

It is noted that one of the current trends in the literature of operations research is the use of some sort of hybridization of different solution techniques. The same is applicable for the Examination Timetabling.

In the following the literatures are divided into the same classification as used for University Course Timetabling in Section 2.3, i.e. based on the main techniques used.

Swarm Intelligence Algorithms

Ant Colony algorithms has been used to solve Examination Timetabling in Dowsland and Thompson (2005) and Eley (2007). In Dowsland and Thompson (2005) an Ant Colony algorithm based on the ANTCOL algorithm for graph coloring from Costa and Hertz (1997) is developed to solve the Toronto benchmarks. A number of enhancements and modifications to the algorithm are introduced. These include an initialization method, using recursive Largest Degree and Saturation Degree, and trail calculations. Eley (2007) compares the ANTCOL algorithm with the MAX-MIN Ant System for University Course Timetabling from Socha et al. (2003). The algorithms are tested on the Toronto benchmarks and it is showed that the ANTCOL system outperforms the MAX-MIN Ant System.

In Turabieh and Abdullah (2011b) the Examination Timetabling problem is solved using the Fish Swarm algorithm developed in Turabieh et al. (2010) for University Course timetabling. The results show that the algorithm performs well on the Toronto benchmark datasets. In later work of the same authors (Turabieh and Abdullah (2011a)) a Great Deluge algorithm within an electromagnetic-like mechanism is employed for Examination Timetabling. This mechanism method shares the same concepts as the Particle Swarm, where the position is changed based on the total force that affects the particle in the search space. The algorithm was tested on the Toronto benchmark and ITC2007 competition datasets.

Sabar et al. (2009) uses a honey bee mating algorithm to solve the Toronto Benchmark datasets. The honey bee mating process is a typically swarm-based approach, were the algorithm is inspired

by the process of the mating of honey bees. The queen (current best solution) leaves the nest to perform a mating flight during which the drones (new solutions) follows the queen and mate with her. In the beginning of the flight, the queen's speed is high and therefore the probability of mating is also high, which is also the case when the fitness of the drones is as good as the queen's. The queen then moves between different solutions in the solution space, according to her speed, and mates with the drones. After a successful mating, a new brood is generated, the fittest replaces the queen, and the rest become the new drones. Solutions show that the honey bee mating algorithm can produce good quality solutions for the Toronto benchmarks.

Evolutionary Algorithms

Wong et al. (2002) present an exam timetable generator which implemented for Ecole de Technologie Superieure at the University of Quebec. The approach of the generator is a Generic Algorithm to construct the exam timetable. A binary tournament selection is used to produce candidates for the algorithm.

Mansour et al. (2011) developed an evolutionary heuristic based on the scatter search approach. Scatter search operates on maintaining and evolving a population of small candidate solutions. It is then possible to find good suboptimal solutions for the problem. The algorithm is compared with other heuristics (genetic algorithm, Simulated Annealing, and 3-phase Simulated Annealing) on real data of the Lebanese American University and the results shows that the adaptive scatter search approach generates the best timetables.

Local Search Techniques

In Casey and Thompson (2003) a Greedy Randomized Adaptive Search Procedure (GRASP) is used for solving the Examination Timetabling problem for the Toronto benchmark datasets. The GRASP algorithm consists of a construction phase, where feasible timetables are created, and an improvement phase. In the constructing phase a limited form of a Tabu Search and efficient ordering of the exams are used. The improvement phase makes use of a neighborhood based on Kempe chains and a limited form of Simulated Annealing. In order to enhance the solution space the algorithm makes use of memory functions. The GRASP algorithm produced is simple to understand and performs robustly across all the instances.

In Ahmadi et al. (2003) a hyper heuristic is developed with a Variable Neighborhood Search to find good combinations of parameterized heuristics. Permutations of twelve low level heuristics are employed to create solutions. Seven of the heuristic are exam selection, two time slot selection and three room selection heuristics. The hyper-heuristic is tested on instances from University of Nottingham.

Merlot et al. (2003) employ constraint programming to create a feasible initial solution for the Examination Timetabling problem. To improve the solution Simulated Annealing and Hill Climbing are used. The hybrid method has been tested on the University of Melbourne, two variants of Toronto instances and the Nottingham benchmark.

Burke et al. (2004a) make used of two variants of local search; a time-predefined variant of Simulated Annealing and an adaptation of the Great Deluge method. The Great Deluge has the same advantage as Simulated Annealing by accepted worse moves during its run. The algorithms are tested on the Toronto and Nottingham benchmarks and it is showed that the Great Deluge approach was superior to the Simulated Annealing approach.

White and Xie (2001) developed a four phase Tabu Search called OTTABU. In each phase more constraints are considered. The OTTABU algorithm was tested on the Examination Timetabling problem at the University of Ottawa, Canada. The approach was extended in White et al. (2004) where the Tabu lists are dynamically relaxed after a certain solution time with no improvements. This extended approach was tested on the Toronto benchmark.

Abdullah et al. (2007) developed a Large Neighborhood Search based on the search methodology originally developed in Ahuja et al. (2001). The key features of the approach are the combination of the very large neighborhood tree-based structure with the technique of identifying improvement

moves by addressing negative cost partition-disjoint cycles. The approach was able to find some new best solutions for the Toronto benchmarks. In Abdullah et al. (2010) a hybridized approach of tabu-based and memetic algorithms is developed. The construction phase of the algorithm is based on a saturation degree graph coloring heuristic and the improvement phase makes use of the hybrid heuristic. A tabu list is used to penalize neighborhood structures that are unable to generate better solutions after the crossover and mutation operators have been applied to the selected solutions from the population pool.

In Caramia et al. (2008) four variant of a local search based algorithm are tested on the Toronto benchmarks and compared with solution from other papers. The algorithm consists of three building blocks; a greedy scheduler to find timetables with small length, a penalty decreaser to reduce the penalty of a schedule without changing the length and a penalty trader to reduce the penalty of a schedule by increasing the number of used time slots. The four variants are different depending on the type of checkpointing used, and on whether bridging priorities were used. The algorithms perform more robust than those of the comparison. The best variation being a hybridization of an approach using adaptive checkpointing and bridging priorities with an approach using constant checkpointing but no bridging priorities.

Pillay and Banzhaf (2009) suggest a hyper-heuristic with hierarchal combination of heuristics. A Tabu Search is used to search the heuristic space for a heuristic list that produces the best quality exam timetable. Each list contains two of the six low level heuristics. The algorithm is tested on the Toronto benchmark and provides all instances with feasible timetables.

Gogos et al. (2012) developed a multi-stage algorithmic process to solve the datasets of ITC2007. The top level heuristic is a GRASP and low level methods consist of several optimization algorithms, heuristics and meta-heuristics. The approach has a construction phase and an improvement phase. Each phase consists of stages that are consumed in a circular fashion.

Graph Coloring Algorithms

Graph bases methods are widely used on Examination Timetabling problems. The algorithms are often hyper-heuristics where the lower level heuristics are graph coloring heuristics, such as largest degree and saturation degree.

In Asmuni et al. (2005) a Fuzzy Logic algorithm was employed to order exams to be scheduled based on graph coloring heuristics. The fuzzy weight of an exam is used to represent how difficult it is to schedule. The method cannot compete with other solution methods on the Toronto benchmark, but the potential of it is demonstrated. Different or more Fuzzy functions are needed to be able to produce best known solutions. In later work of the same authors, a Fuzzy system was developed to evaluate the quality of the exam timetables (Asmuni et al. (2007)). The quality of exam timetables is measured considering two criteria: the average penalty per student and the highest penalty imposed on any of the students.

Yang and Petrovic (2005) used a hyper-heuristic with a Case-Based Reasoning as high level heuristics to choose graph heuristics to construct a feasible initial solution. A Great Deluge algorithm is then employed to improve the solution. Burke et al. (2005) also make use of a Case-Based Reasoning in a hyper-heuristic. Two different ways of hybridizing the low level graph heuristics were compared for solving the Toronto benchmark data. One were Case-Based Reasoning is used as higher level heuristic and one with a Tabu Search. The Tabu Search approach performed a little better than the Case-Based Reasoning approach, however it was significant slower.

The Tabu Search as higher level heuristic for low level graph coloring heuristics were investigated once more in Burke et al. (2007). It was observed that the more different graph-based heuristics used in the lower level the better the performance might by. The drawback is however the enlargement of the search space which can influence the solving time. The approach was used on both the Examination Timetabling and University Course Timetabling and in both cases it was competitive with other solution methods from the literature.

The Examination Timetabling problem from Universiti Malaysia Pahang, Malaysia, is presented in Kahar and Kendall (2010). The paper compares the Examination Timetabling problem at

the university with known benchmark data, and two new constraint concerning splitting exams into different rooms are introduced. When splitting an exam the rooms must be in the same building (hard constraint) and the distance between the rooms should be as close as possible (soft constraint). The algorithm presented in the paper is based on graph coloring heuristics and produce superior solutions compared to the existing software at the university.

Sabar et al. (2012) developed a graph coloring constructive hyper-heuristic algorithm. At the higher level of the hyper heuristic the difficulty level of examinations is calculated by using hybridizations of four graph coloring heuristics. At the low level four graph coloring algorithms are used; largest degree, saturation degree, largest colored degree and largest enrollment. The results show that the approach is a simple and an efficient tool to produce competitive results for the Toronto and the ITC2007 benchmark data.

Rahman et al. (2014) employ adaptive linear combinations of graph coloring heuristics with a heuristic modifier to address the Examination Timetabling problem for the Toronto and the ITC2007 benchmark data. The approach makes use of two graph coloring heuristics, largest descent and saturation degree.

Other Heuristic Methods

This section contains literature which application method does not fit in the previous heuristic categories.

In Petrovic and Bykov (2003) the multi-criteria called compromised programming is used. The method requires the user to specify a reference solution. To improve the values of the reference objectives a trajectory is drawn in the criteria space and a Great Deluge is conducted using the specified trajectory. The criteria weights can be dynamically changed to guide the search, starting from random points, towards the reference point.

Duong and Lam (2004) present a two phase heuristic for the Examination Timetabling problem. In the first phase constraint programming is used to generate a feasible initial solution and in the second phase a Simulated Annealing with Kempe chain neighborhood improves that solution. The algorithm is tested on real data of Ho Chi Minh City University of Technology.

Ozcan et al. (2009) uses a late acceptance hyper-heuristic to solve the Toronto benchmark data. Late acceptance strategy is a memory based technique that maintains the history of objective values from the last $\mathcal{L}$ previous solutions. The new solution is compared to a previous solution obtained at the $\mathcal{L}$'th step and the acceptance decision is made accordingly. The results show that Simple Random performs the best when combined with late acceptance as compared with the other heuristic of the paper.

Abdullah et al. (2009) present a hyper-heuristic of an electromagnetic-like mechanism and the Great Deluge algorithm. Electromagnetic-like mechanism is implemented to calculate the force for each solution. The force value later will be used in the Great Deluge algorithm to calculate the force decay rate.

Another hyper-heuristic for Examination Timetabling is the Monte Carlo based hyper heuristic developed in Burke et al. (2010a). A number of new and previous suggested Monte Carlo based selection hyper-heuristics are investigated on the Toronto benchmark data. As heuristic selection methods a simple random algorithm, a greedy algorithm, a Choice Function algorithm and a learning scheme are utilized. The hyper-heuristic make use of four low level heuristics.

In Demeester et al. (2012) a tournament based hyper-heuristic is presented. The hyper-heuristic framework of Özcan et al. (2008) is extended using a *tournament factor*. At each iteration, the selected heuristic generates a predefined number, namely the tournament factor, of random moves. The low level heuristics are Simulated Annealing, Great Deluge and steepest descent. The hyper heuristic is tested on the Toronto benchmark, the instances from ITC2007 and datasets from KAHO Sint-Lieven, Belgium.

Anwar et al. (2013) propose a harmony search based hyper-heuristic method for the Examination Timetabling. The harmony search algorithm is a fairly new meta-heuristic method inspired by musical improvisation process. The algorithm of this paper employed harmony search algorithm at the high level to evolve a sequence of improvement low-level heuristics. At the low level, two

different neighborhood structures are used. Swap and move. The algorithm is tested on the ITC2007 benchmark

Exact Methods

The amount of research using some sort of exact methods, such as decomposition, is quite sparse for Examination Timetabling.

In Qu and Burke (2007) a new decomposition technique is developed for the Examination Timetabling. The idea is to decompose the problem into two sub-sets of events; a difficult and an easy sub-set. In the first step the exams of the difficult set is ordered to find the best feasible solution. In second step the solution from step one is fixed and the easy events are ordered. The two steps are then repeated in a cycle. The approach was tested on the Toronto benchmark data.

Al-Yakoob et al. (2010) considers two exam related problems at Kuwait University; the Examination Timetabling problem and the proctor assignment problem. A Mixed Integer Programming model has been created for both problem and solved using the State-of-art MIP solver CPLEX. The results obtained by solving the MIP models are of significant improvements compared to the existing manual approach at the given university.

Software Systems

Many of the systems mentioned in Section 2.3.3 are for both University Course Timetabling and Examination Timetabling. In Dimopoulou and Miliotis (2001) the initial solution for the Examination Timetabling is based on the University Course Timetabling. This is then adjusted repeatedly by a heuristic approach.

The newest paper from the system UniTime on Examination Timetabling is Müller (2013). The algorithm presented consists of several phases. A construction phase where a complete solution is found using an Iterative Forward Search (Müller et al. (2005)). The next phase uses a Hill Climber to find a local optimum. Once a solution cannot be improved further a Great Deluge technique is used.

Other systems which only solves the Examination Timetabling problem includes Hansen and Vidal (1995) and Thompson and Dowsland (1998).

Hansen and Vidal (1995) is a system for timetabling oral and written examinations at more than 200 high schools in Denmark. The system uses a four phase process dealing with different objectives using different techniques. Phase one is the subject draft which determine which exams a student is assigned. Examination chains are generated in phase two. Phase three creates the examination timetables and finally phase four assigns censorships to the exams.

Thompson and Dowsland (1998) create an Examination Timetabling system at Swansea University in Wales. The problem is divided in two phases both solved using Simulated Annealing. The first phase seeks out a feasible solution and the second finds an improvement in terms of meeting the secondary objectives and soft constraints.

All the references mentioned in this section are listed in Table 2.6 in 2.A.

2.6 Student Sectioning

The previous mentioned educational timetabling problems are all considering the problem of assigning some events to times. *Student Sectioning* usually resides outside this categorization as it involves assigning students to sections and not times.

A course might be split into sections/classes, i.e. copies of the same course, each with its own time, room and teachers. Student Sectioning is the problem of assigning students to sections of courses while respecting the requests of the individual student. Some of primary hard and soft constraints of Student Sectioning problems are listed below:

- Primary hard constraints
  - No 1-order conflicts. A student cannot attend two courses that are overlapping in time.
  - Limitations on class sizes.
  - Resource limitation. E.g. only two Physic classes in each cluster.
- Primary soft constraints
  - Equally distribution between sections of same course (spreading)
  - Minimize the number of sections used.

2.6.1 Recent Research

Literature concerning Student Sectioning is very sparse, none of the previous surveys listed in Section 2.1.1 use much effort on this research subject and in many papers where Student Sectioning is mentioned the papers are on University Course Timetabling or High School Timetabling and not specific on Student Sectioning.

The surveys of Carter and Laporte (1998) and Schaerf (1999a) provides a good overview of previous work within practical course timetabling and automated timetabling problems. Both papers give a short description of student sectioning problem and some of the earliest work on the subject. Kingston (2013a) makes a short description on the subject, however the book chapter only reefers to few articles. Other surveys only briefly mentioned that many articles on timetabling problems have a preprocessing problem of assigning students to classes.

Student Sectioning arises at both universities and high schools. In both cases it is often a preprocessing problem for the timetabling problem.

Of the two cases, Student Sectioning at high schools is the least studied and are in general smaller in size compared to university student sectioning. In de Haan et al. (2007) *optional subject* for the students is used when constructing timetables at high schools in Netherlands, and *cluster schemes* are created to maintain the students' optional courses. Due to a new educational system at the test case high school the program in the paper is only used operational for the Student Sectioning (constructing the cluster schemes) and not on the timetabling.

Kristiansen and Stidsen (2013) use the comparative term *electives*. The problem is to assign 2nd and 3rd year students to electives given their requests while minimizing the number of classes created and producing a fair distribution. An Adaptive Large Neighbor Search algorithm has been created to solve the problem and the results are in average 0.5% from the best known lower bound. The algorithm developed in Kristiansen and Stidsen (2013) is implemented in the cloud-based high school administration software *Lectio*.

In Kristiansen et al. (2013a) the problem is concerned the first year students at Danish high schools and is bipartite. First the students are grouped into cohorts in which they are going to attend the same mandatory courses, secondly the cohorts are assigned time slots to satisfy the students requests for two electives. The problem is solved using the MIP solver Gurobi.

There exist more papers on Student Sectioning at universities, however in many cases you must search in papers on University Course Timetabling to find discussion on the subject, e.g Rudova and Murray (2003) and Suyanto (2010).

Carter (2001) and Rudova and Murray (2003) describe a *demand-driven timetabling* where student selections of courses are utilized to create a timetable that satisfy as many requests as possible. In Carter (2001) the University Course Timetabling is created first by assigning time slots to sections and the students is then assigned individually to the sections that maximizes timetable satisfaction and balance section sizes. Rudova and Murray (2003) uses student course selections to construct timetables that attempt to maximize the number of satisfied course requests.

In Sönmez and Ünver (2010) they make use of a bidding system. The students make bids on which courses they want to participate in, and based on these bids the courses are placed in

class rooms which size reflects the number of bids for the given course. This paper looks on the mechanisms within the course bidding.

The approach of using "bidding/requests" and solving Student Sectioning as a subproblem to the university course timetabling is the most common method. (Aubin and Ferland (1989); Sampson et al. (1995); Robert and Hertz (1996))

In other papers the students are clustered to avoid conflicts between the students' choices of courses. Banks et al. (1998) formulates the timetabling problem as a *Constraint Satisfaction Problem* (CSP) where the algorithm iteratively adds subset constraints to the CSP formulation. Amintoosi and Haddadnia (2005) proposed a fuzzy clustering algorithm to create an initial sectioning prior to timetabling a set of classes. The same approach is used in Alvarez-Valdes et al. (2000). The students select course in an interactive process in the first phase and in the second phase a Tabu search algorithm is used for constructing the timetable.

In Müller and Murray (2010) Student Sectioning is solved as a part of the University Course Timetabling where it is considered during and after the creation of the timetabling. In Suyanto (2010) used a two stage approach for solving the university course timetabling, where batch student sectioning is done by allowing the first stage timetable to change.

All the references mentioned in this section are listed in Table 2.7 in 2.A.

2.7 Conclusion

This paper has been provided in in-depth survey on educational timetabling literature in the last decade. The survey provides a comprehensive overview of methodologies used for each of the four main subjects within educational timetabling problems; University Course Timetabling, High School Timetabling, Examination Timetabling and Student Sectioning. For each planning problem a description is given along with benchmarks and recent research.

It is possible to draw a few conclusions for the complete educational timetabling problems. Firstly, within the last decade the amount of successful literature on this subject has been increasingly and many of the used solution approach is of some kind of hybridization of multiple heuristics.

Secondly, in many cases the quality of a solution is only compared to previous solutions and not on the optimal or lower bound. The main approach is some sort of heuristic, it could be advantage to research in the use of more exact methods to create some good lower bounds or better yet, optimal solutions to the benchmark data.

Finally, there is still a problem of closing the gap between theory and practice. The different planning problems are still in need of some generalized format and description and more benchmark data from the real world. The XHSTT format for High School Timetabling is an excellent example on a generalized description on a educational planning problem with corresponding benchmark consisting of real world data from a range of countries.

2.A Summary Tables

This section contains summarization tables on the literature mentioned in each section, listed in order of appearance in this survey. For each section, the references are sorted according to the year of publication. Some of the mentioned papers in the tables might cover more than one subject and these papers are therefore listed in all the corresponding tables.

Each table consists of the description of author(s), research area and comments.

First we have the tables related to previous surveys and competitions within educational timetabling in Table 2.1 and Table 2.2, respectively.

Table 2.7 lists all the papers related to Student Sectioning.

Table 2.1: Surveys on educational timetabling

References and publication year	Title	Research area	Comments
Schmidt and Ströhlein (1980)	Timetable construction – an annotated bibliography	Timetabling	Provide an annotated bibliography including more than 200 entries.
de Werra (1985)	An introduction to timetabling	Class-teacher and course timetabling	Graph coloring and network flows methods.
Junginger (1986)	Timetabling in Germany–A Survey	School timetabling	Various software products implemented in Germany.
Carter (1986)	A Survey of Practical Applications of Examination Timetabling Algorithms	Examination timetabling	Algorithms tested on real data or implemented.
Carter and Laporte (1996)	Recent developments in practical examination timetabling	Examination timetabling	Algorithms tested on real data or implemented from 1986 to 1996.
Bardadym (1996)	Computer-aided school and university timetabling: The new wave	Educational timetabling	Computer-aided management systems for timetabling.
Wren (1996)	Scheduling, timetabling and rostering — A special relationship?	Scheduling and timetabling	Links scheduling, timetabling and rostering.
Carter and Laporte (1998)	Recent developments in practical course timetabling	Course timetabling	Algorithms tested on real data or implemented.
Schaerf (1999a)	A Survey of Automated Timetabling	School, course and exam timetabling	Classification of solution techniques particularly from artificial intelligence.
Burke and Petrovic (2002)	Recent research directions in automated timetabling	University timetabling (course and exam)	Tries to explore approaches that can operate at a higher level of generality.
Burke et al. (2004c)	Application to timetabling	Class-teacher, course, exam and sport timetabling	Application of graph coloring methods to timetabling.
McCollum (2006)	University Timetabling: Bridging the Gap between Research and Practice	Latest survey on University Course Timetabling.	
Qu et al. (2009)	A Survey of Search Methodologies and Automated System Development for Examination Timetabling	Examination timetabling	Algorithms from 1996 to 2009. Presenting a definitive renaming of different benchmark problem datasets.
Pillay (2013)	An Overview of School Timetabling Research	School timetabling	A standardized definition of the problem in terms of problem requirements, hard constraints and soft constraints.
Kingston (2013a)	Educational Timetabling	-	Book chapter introducing educational timetabling problems.

Table 2.2: Competitions within Educational Timetabling

References	Research area	Comments
Kostuch (2004)	University Timetabling	Winner of the 1st International Timetabling Competition 2002
Gaspero et al. (2007)	University Timetabling and Exam Timetabling	Overview of the competitions and the competitors of the 2nd International Timetabling Competition 2007
McCollum et al. (2010)	Setting the Research Agenda in Automated Timetabling: The Second International Timetabling Competition	
Post et al. (2012c)	High School Timetabling	Overview of the competitions and the competitors of the 3rd International Timetabling Competition 2011

Table 2.3: University Course Timetabling

Authors	Problem	Comments
Carter (2001)	A comprehensive course timetabling and student scheduling system at the university of waterloo	Implemented at University of Waterloo and in 2001 it has been used successfully for 15 years.
Dimopoulou and Miliotis (2001)	Implementation of a university course and examination timetabling system	A system for University Course Timetabling and Examination Timetabling at Athens University of Economics and Business.
Socha et al. (2003)	Ant Algorithms for the University Course Timetabling Problem with Regard to the State-of-the-Art	-
Rudova and Murray (2003)	University Course Timetabling with Soft Constraints	A part of the UniTime systems publications. Using constraint logic programming.
Gaspero and Schaerf (2003)	Multi-neighbourhood Local Search with Application to Course Timetabling	Tested on the School of Engineering at Udine University.
Martin (2004)	Ohio University's College of Business Uses Integer Programming to Schedule Classes	Solves the University Course Timetabling at College of Business, Ohio University.
Daskalaki et al. (2004)	An integer programming formulation for a case study in university timetabling	Engineering School of Greek universities.
Kostuch (2004)	Timetabling Competition - SA-based Heuristic	Winner of ITC2002
Müller et al. (2005)	Minimal Perturbation Problem in Course Timetabling	Iterative Forward Search. Part of UniTime system.
Avella and Vasil'Ev (2005)	A Computational Study of a Cutting Plane Algorithm for University Course Timetabling	Branch-and-Cut method with two cutting planes.
Daskalaki and Birbas (2005)	Efficient solutions for a university timetabling problem through integer programming	A two stage relaxation procedure.

Continued on next page

Table 2.3 – continued from previous page

Authors	Problem	Comments
Qualizza and Serafini (2005)	A Column Generation Scheme for Faculty Timetabling	Column Generation with a Branch-and-Bound method to ensure feasibility.
Chiarandini et al. (2006)	An effective hybrid algorithm for university course timetabling	Tested on the benchmark of ITC2003.
Lewis et al. (2007)	Post Enrolment based Course Timetabling: A Description of the Problem Model used for Track Two of the Second International Timetabling Competition	Explanation of Post enrollment-based course timetabling used for ITC2007 Track 3.
Al-Yakoob and Sherali (2007)	A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations	Incorporates gender policies in the University Course Timetabling at Kuwait University.
Burke et al. (2007)	A graph-based hyper-heuristic for educational timetabling problems	Tested on the benchmark of ITC2007.
Murray et al. (2007)	Modeling and Solution of a Complex University Course Timetabling Problem	Transform the University Course Timetabling problem at Purdue University into a constraint satisfaction and optimization problem. Part of the UniTime system.
Schimmelpfeng and Helber (2007)	Application of a real-world university-course timetabling model solved by integer programming	Solves the problem at School of Economics and Management at Hannover University.
Lach and Lübbecke (2008)	Optimal University Course Timetables and the Partial Transversal Polytope	Theoretical background for a two stage decomposition method for the University Course Timetabling.
Nurmi and Kyngas (2008)	A Conversion Scheme for Turning a Curriculum-Based Timetabling Problem into a School Timetabling Problem	Transform the problem description of Curriculum-Based Timetabling Problem from ITC2007 to a school timetabling problem and solves it using a genetic algorithm.
Cambazard et al. (2008)	Local Search and Constraint Programming for the Post Enrolment-based Course Timetabling Problem	Winner of ITC2007 Track 2.
Müller (2009)	ITC2007 solver description: a hybrid approach	Winner of ITC2007 Track 1 and 3.
Suyanto (2010)	An informed genetic algorithm for university course and student timetabling problems	-
Turabieh et al. (2010)	Fish Swarm Intelligent Algorithm for the Course Timetabling Problem	Applied to the definition of Socha et al. (2003).
Lü et al. (2011)	Neighborhood analysis: a case study on curriculum-based course timetabling	-
Rudová et al. (2011)	Complex university course timetabling	Newest research paper from UniTime on Post Enrollment-based Course Timetabling.
Nguyen et al. (2011)	Variable Neighborhood Search for a Real-World Curriculum-Based University Timetabling Problem	Solves 14 instances from University of Science in Vietnam.

Continued on next page

Table 2.3 – continued from previous page

Authors	Problem	Comments
Shiau (2011)	A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences	Solves the course timetabling at a university in Taiwan.
Hao and Benlic (2011)	Lower bounds for the ITC-2007 curriculum-based course timetabling problem	Generates lower bounds for the ITC2007 instances.
Bonutti et al. (2012)	Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results	Formulations and collection of the Udine benchmark.
Lach and Lübbecke (2012)	Curriculum based course timetabling: new solutions to Udine benchmark instances	Uses the two stage decomposition of Lach and Lübbecke (2008) to solve the Udine Benchmark.
Müller and Rudová (2012)	Real-life curriculum-based timetabling	Newest research paper from UniTime on Curriculum-based Course Timetabling.
Burke et al. (2012)	A branch-and-cut procedure for the Udine Course Timetabling problem	Generates good lower bounds fast.
Ceschia et al. (2012)	Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem	Tested on the ITC2003 and ITC2007 benchmark data.
Nothegger et al. (2012)	Solving the post enrolment course timetabling problem by ant colony optimization	Tested on the ITC2007 benchmark with mixed results.
Chen and Shih (2013)	Solving University Course Timetabling Problems Using Constriction Particle Swarm Optimization with Local Search	-
Cacchiani et al. (2013)	A new lower bound for curriculum-based course timetabling	Lower bounds on the benchmark sets of ITC2003 and ITC2007.
Azlan and Hussin (2013)	Implementing graph coloring heuristic in construction phase of curriculum-based course timetabling problem	Tested on the benchmark of ITC2007.

Table 2.4: High School timetabling - XHSTT

Authors	Problem	Comments
Post et al. (2012a)	An XML format for benchmarks in High School Timetabling	The XHSTT format description.
Fonseca et al. (2012)	A SA-ILS approach for the High School Timetabling Problem	1st place at ITC2011.
Kheiri et al. (2012)	HySST: Hyper-heuristic Search Strategies and Timetabling	2nd place at ITC2011.
Sørensen et al. (2012).	International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm	3rd place at ITC2011.

Continued on next page

Table 2.4 – continued from previous page

Authors	Problem	Comments
Romrös and Homberger (2012)	An Evolutionary Algorithm for High School Timetabling	4th place at ITC2011.
Post et al. (2013)	The Third International Timetabling Competition	Description and results of ITC2011.
Pimmer and Raidl (2013)	A Timeslot-Filling Heuristic Approach to Construct High-School Timetables	
Fonseca et al. (2013)	Late Acceptance-Hill Climbing Applied to the High School Timetabling Problem	Improvements of Fonseca et al. (2012).
Kristiansen et al. (2013c)	Integer Programming for the Generalized (High) School Timetabling Problem	IP model for the XHSTT format and creation of lower bounds.

Table 2.5: High School timetabling - country based

Studies	Subject	Comments
Wood and Whitaker (1998)	Student Centred School Timetabling	Solves the problem at Secondary schools in New Zealand using Hill Climber and Hungarian Assignment.
Colorni et al. (1998)	Metaheuristics for High School Timetabling	Solves two Italian high school instances. A genetic algorithm with Tabu Search being the best.
Schaerf (1999b)	Local search techniques for large high school timetabling problems	Solves one artificial and two Italian high schools using a hybrid heuristic with Tabu Search.
Fernandes et al. (1999)	High school weekly timetabling by evolutionary algorithms	High schools in Portugal.
Bufé et al. (2001)	Automated Solution of a Highly Constrained School Timetabling Problem - Preliminary Results	Using Tabu Search to solve the German high school timetabling problem.
Filho and Lorena (2001)	A Constructive Evolutionary Approach to School Timetabling	Solves two Brazilian high schools.
Lohnertz (2002)	A timetabling system for the German gymnasium	Combines Tabu Search and Graph Vertex Coloring for German high school.
Wilke et al. (2002)	A Hybrid Genetic Algorithm for School Timetabling	German high school.
Souza et al. (2003)	A GRASP-tabu search algorithm for school timetabling problems	Brazilian high schools.
Valouxis and Housos (2003)	Constraint programming approach for school timetabling	Greek high schools.
Papoutsis et al. (2003)	A column generation approach for the timetabling problem of Greek high schools	Greek high schools.

Continued on next page

Table 2.5 – continued from previous page

Studies	Subject	Comments
Santos et al. (2004)	An Efficient Tabu Search Heuristic for the School Timetabling Problem	Brazilian high schools.
Post and Ruizenaar (2004)	Clusterschemes in Dutch secondary schools	Clusterschemes are constructed and a Branch-and-Bound approach is then used for Dutch high school timetabling problem.
Kingston (2005)	A Tiling Algorithm for High School Timetabling	Solves seven instances of Australian high schools.
Santos et al. (2005)	A Tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem	Uses memory based diversifications to improve robustness in a Tabu Search. Brazilian high schools.
Jacobsen et al. (2006)	Timetabling at German Secondary Schools: Tabu Search versus Constraint Programming	German high school.
Melício et al. (2006)	THOR: A Tool for School Timetabling	Describes the system THOR which in use by more than 100 schools in Portugal.
Yigit (2007)	Constraint-Based School Timetabling Using Hybrid Genetic Algorithms	Solves the problem at the Technical and Vocational High Schools in Turkey.
Nurmi and Kyngas (2007)	A framework for school timetabling problem	Finnish high schools.
de Haan et al. (2007)	A Case Study for Timetabling in a Dutch Secondary School	Solves the Dutch case from Post and Ruizenaar (2004) using a three phase approach.
Avella et al. (2007)	A computational study of local search algorithms for Italian high-school timetabling	Using a hybrid heuristic with Variable Neighborhood Search and Simulated Annealing on two Italian high schools.
Bello et al. (2008)	An Approach for the Class/Teacher Timetabling Problem using Graph Coloring	Tested on five Brazilian high schools.
Boland et al. (2008)	New integer linear programming approaches for course timetabling	Australian high schools.
Beligiannis et al. (2008)	Applying evolutionary computation to the school timetabling problem: The Greek case	Greek high schools made available as benchmark data.
Moschopoulos et al. (2009)	A User-Friendly Evolutionary Tool for High-School Timetabling	An user interface for the system created in Beligiannis et al. (2008).
Beligiannis et al. (2009)	A genetic algorithm approach to school timetabling	Solves the Beligiannis benchmark.
Raghavjee and Pillay (2009)	Evolving solutions to the school timetabling problem	Solves the Beligiannis benchmark.
Birbas et al. (2009)	School timetabling for quality student and teacher schedules	Solves a secondary Hellenic school. First by assigning teachers to shifts and then solving the High School Timetabling problem.

Continued on next page

Table 2.5 – continued from previous page

Studies	Subject	Comments
Moura and Scaraficci (2010)	A GRASP strategy for a more constrained School Timetabling Problem	Solves the Brazilian high schools Teacher-class assignment problem.
Zhang et al. (2010)	A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems	Tested on two set of benchmark instances. A randomly generated and instances from Greek high schools in Patras.
Santos et al. (2012)	Strong bounds with cut and column generation for class-teacher timetabling	Brazilian high schools Class-teacher assignment problem.
Raghavjee and Pillay (2010)	An informed genetic algorithm for the high school timetabling problem	High schools in South Africa.
Minh et al. (2010)	Using Tabu Search for Solving a High School Timetabling Problem	High schools in Vietnam solved using Tabu Search.
Post et al. (2012b)	Cyclic transfers in school timetabling	High school timetabling in the Netherlands and England.
Ribic and Konjicija (2010)	A two phase integer linear programming approach to solving the school timetable problem	Croatian secondary school.
Valouxis et al. (2012)	Decomposing the High School Timetable Problem	Greek high schools.
Sørensen and Stidsen (2013)	Integer Programming and Adaptive Large Neighborhood Search for Real-World Instances of High School Timetabling	Implemented at available for more than 200 Danish high schools.
Sørensen and Dahms (2014)	A Two-Stage Decomposition of High School Timetabling applied to cases in Denmark	Using the approach of Lach and Lübbecke (2012) on the Danish high school timetabling problem.

Table 2.6: Examination timetabling

References	Problem	Comments
Hansen and Vidal (1995)	Planning of high school examinations in Denmark	System for examination timetabling and censorships assignment at Danish high schools.
Carter et al. (1996)	Examination Timetabling: Algorithmic Strategies and Applications	Created the Toronto benchmark datasets.
Burke et al. (1996)	A Memetic Algorithm for University Exam Timetabling	Extended Carter et al. (1996) with the Nottingham benchmark datasets.
Thompson and Dowsland (1998)	A robust simulated annealing based examination timetabling system	A system for handling exams at Swansea University, Wales.
White and Xie (2001)	Examination Timetables and Tabu Search with Longer-Term Memory	University of Ottawa, Canada.

Continued on next page

Table 2.6 – continued from previous page

References	Problem	Comments
Dimopoulou and Miliotis (2001)	Implementation of a university course and examination timetabling system	A system for University Course Timetabling and Examination Timetabling at Athens University of Economics and Business.
Wong et al. (2002)	Final exam timetabling: a practical approach	Solve the examination problem at Ecole de Technologie Superieure, Montreal, Canada, and has been in use since 2001.
Ahmadi et al. (2003)	Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem	Tested on the Nottingham benchmark.
Merlot et al. (2003)	A Hybrid Algorithm for the Examination Timetabling Problem	Created the Melbourne benchmarks.
Petrovic and Bykov (2003)	A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories	Tested on the Nottingham and Toronto benchmarks.
Casey and Thompson (2003)	GRASPing the Examination Scheduling Problem	Tested on the Toronto benchmarks.
Burke et al. (2004a)	A time-predefined local search approach to exam timetabling problems	Tested on the Nottingham and Toronto benchmarks.
Duong and Lam (2004)	Combining Constraint Programming and Simulated Annealing on University Exam Timetabling	HoChiMinh City University of Technology, Vietnam.
White et al. (2004)	Using tabu search with longer-term memory and relaxation to create examination timetables	University of Ottawa, Canada.
Burke et al. (2005)	Hybrid Graph Heuristics within a Hyper-Heuristic Approach to Exam Timetabling Problems	Tested on the Toronto benchmarks.
Asmuni et al. (2005)	Fuzzy Multiple Heuristic Orderings for Examination Timetabling	Tested on the Toronto benchmarks.
Yang and Petrovic (2005)	A Novel Similarity Measure for Heuristic Selection in Examination Timetabling	Tested on the Toronto benchmarks.
Dowsland and Thompson (2005)	Ant colony optimization for the examination scheduling problem	Tested on the Toronto benchmarks.
Abdullah et al. (2007)	Investigating Ahuja–Orlin's large neighbourhood search approach for examination timetabling	Tested on the Toronto benchmarks.
Asmuni et al. (2007)	A Novel Fuzzy Approach to Evaluate the Quality of Examination Timetabling	Tested on the Toronto benchmarks.
Burke et al. (2007)	A graph-based hyper-heuristic for educational timetabling problems	Tested on the Toronto benchmarks.
Eley (2007)	Ant Algorithms for the Exam Timetabling Problem	Tested on the Toronto benchmarks.

Continued on next page

Table 2.6 – continued from previous page

References	Problem	Comments
Qu and Burke (2007)	Adaptive decomposition and construction for examination timetabling problems	Tested on the Toronto benchmarks.
Caramia et al. (2008)	Novel Local-Search-Based Approaches to University Examination Timetabling	Tested on the Toronto benchmarks.
Pillay and Banzhaf (2009)	A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem	Tested on the Toronto benchmarks.
Ozcan et al. (2009)	Examination timetabling using late acceptance hyper-heuristics	Tested on the Toronto benchmarks.
Abdullah et al. (2009)	A Hybridization of Electromagnetic-Like Mechanism and Great Deluge for Examination Timetabling Problems	Tested on the Toronto benchmarks.
Müller (2009)	ITC2007 solver description: a hybrid approach	Winner of ITC2007 Track 1 and 3.
Sabar et al. (2009)	Solving Examination Timetabling Problems using Honey-bee Mating Optimization	Tested on the Toronto benchmarks.
Burke et al. (2010a)	Monte Carlo hyper-heuristics for examination timetabling	Tested on the Toronto benchmarks.
Abdullah et al. (2010)	A Tabu-Based Memetic Approach for Examination Timetabling Problems	Tested on the Toronto benchmarks.
Kahar and Kendall (2010)	The examination timetabling problem at Universiti Malaysia Pahang: Comparison of a constructive heuristic with an existing software solution	Universiti Malaysia Pahang, Malaysia.
Al-Yakoob et al. (2010)	A mixed-integer mathematical modeling approach to exam timetabling	Kuwait University.
Mansour et al. (2011)	Scatter search technique for exam timetabling	Lebanese American University, Lebanon.
Turabieh and Abdullah (2011a)	An integrated hybrid approach to the examination timetabling problem	Tested on Toronto benchmarks and the datasets from ITC2007.
Turabieh and Abdullah (2011b)	A Hybrid Fish Swarm Optimisation Algorithm for Solving Examination Timetabling Problems	Tested on the Toronto benchmarks.
Sabar et al. (2012)	A graph coloring constructive hyper-heuristic for examination timetabling problems	Tested on Toronto benchmarks and the datasets from ITC2007.
McCollum et al. (2012)	A new model for automated examination timetabling	Tested on the datasets from ITC2007.
Gogos et al. (2012)	An improved multi-staged algorithmic process for the solution of the examination timetabling problem	Tested on Toronto benchmarks and the datasets from ITC2007.

Continued on next page

Table 2.6 – continued from previous page

References	Problem	Comments
Demeester et al. (2012)	A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice	Tested on the Toronto benchmark, the ITC 2007 benchmarks and the examination timetabling problem at KAHO Sint-Lieven (Ghent, Belgium).
Müller (2013)	Real-life Examination Timetabling	Purdue University datasets.
Anwar et al. (2013)	Harmony Search-based Hyper-heuristic for examination timetabling	ITC2007 benchmark.
Rahman et al. (2014)	Adaptive linear combination of heuristic orderings in constructing examination timetables	Tested on Toronto benchmarks and the datasets from ITC2007.

Table 2.7: Student sectioning

Author(s)	Problem	Comments
Aubin and Ferland (1989)	A large scale timetabling problem	Tested on data from a High School in Montreal, Canada.
Sampson et al. (1995)	Class scheduling to maximize participant satisfaction	Using a local search heuristic and is able to meet 94% of the requirements at the Graduate School of Business Administration at the University of Virginia, USA.
Robert and Hertz (1996)	How to decompose constrained course scheduling problems into easier assignment type subproblems	Course timetabling with student requests.
Banks et al. (1998)	A heuristic incremental modeling approach to course timetabling	Constraints are added to the timetabling model to avoid student conflicts. The students are individually scheduled after complete timetabling.
Alvarez-Valdes et al. (2000)	Assigning students to course sections using tabu search	Used at the Faculty of Mathematics, University of Valencia, Spain of the academic year 96/97.
Carter (2001)	A comprehensive course timetabling and student scheduling system at the university of waterloo	Implemented at University of Waterloo and in 2001 it has been used successfully for 15 years.
Rudova and Murray (2003)	University course timetabling with soft Constraints	Tested on a data for the fall semester 2001 at Purdue University, USA.
Amintoosi and Haddadnia (2005)	Feature selection in a fuzzy student sectioning algorithm	Tested on the Mathematical department of Sabzevar University, Iran.
de Haan et al. (2007)	A case study for timetabling in a Dutch secondary school	Is used operationally only for constructing the cluster schemes.
Sönmez and Ünver (2010)	Course bidding at business schools	Mechanisms within course bidding.

Continued on next page

Table 2.7 – continued from previous page

Author(s)	Problem	Comments
Müller and Murray (2010)	Comprehensive approach to student sectioning for Purdue University, USA	Student sectioning during course timetabling and batch sectioning after a complete timetable. Implemented in the open source software UniTime.
Suyanto (2010)	An informed genetic algorithm for university course and student timetabling problems	Batch student sectioning within a sparse timetabling model.
Kristiansen and Stidsen (2013)	Elective course student sectioning	Implemented and used for the majority of the Danish high schools from 2012 and forward.
Kristiansen et al. (2013a)	High school student sectioning at Danish high schools	Tested on 25 real life instances from Danish high schools.

Bibliography

S. Abdullah, S. Ahmadi, E. Burke, and M. Dror. Investigating ahuja–orlin's large neighbourhood search approach for examination timetabling. *OR Spectrum*, 29:351–372, 2007. ISSN 0171-6468. 10.1007/s00291-006-0034-7.

S. Abdullah, H. Turabieh, and B. McCollum. A hybridization of electromagnetic-like mechanism and great deluge for examination timetabling problems. In M. J. Blesa, C. Blum, L. Gaspero, A. Roli, M. Sampels, and A. Schaerf, editors, *Hybrid Metaheuristics*, volume 5818 of *Lecture Notes in Computer Science*, pages 60–72. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-04917-0.

S. Abdullah, H. Turabieh, B. McCollum, and P. McMullan. A tabu-based memetic approach for examination timetabling problems. In J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, editors, *Rough Set and Knowledge Technology*, volume 6401 of *Lecture Notes in Computer Science*, pages 574–581. Springer Berlin / Heidelberg, 2010.

S. Ahmadi, R. Barrone, P. Cheng, P. Cowling, and B. McCollum. Perturbation based variable neighbourhood search in heuristic space for examination timetabling problem. In *Multidisciplinary International Scheduling: Theory and Application (MISTA 2003)*, pages 155–171, August 2003.

R. K. Ahuja, J. B. Orlin, and D. Sharma. Multi-exchange neighborhood structures for the capacitated minimum spanning tree problem. *Mathematical Programming*, 91(1):71–97, 2001. ISSN 0025-5610.

S. Al-Yakoob, H. D. Sherali, and M. Al-Jazzaf. A mixed-integer mathematical modeling approach to exam timetabling. *Computational Management Science*, 7(1):19–46, 2010. ISSN 1619-697X.

S. M. Al-Yakoob and H. D. Sherali. A mixed-integer programming approach to a class timetabling problem: A case study with gender policies and traffic considerations. *European Journal of Operational Research*, 180(3):1028 – 1044, 2007. ISSN 0377-2217.

R. Alvarez-Valdes, E. Crespo, and J. M. Tamarit. Assigning students to course sections using tabu search. *Annals of Operations Research*, 96(1-4):1–16, 2000. ISSN 0254-5330.

M. Amintoosi and J. Haddadnia. Feature selection in a fuzzy student sectioning algorithm. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 147–160. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

K. Anwar, A. Khader, M. Al-Betar, and M. Awadallah. Harmony search-based hyper-heuristic for examination timetabling. In *IEEE 9th International Colloquium on Signal Processing and its Applications (CSPA), 2013*, pages 176–181, 2013.

H. Asmuni, E. Burke, J. Garibaldi, and B. McCollum. Fuzzy multiple heuristic orderings for examination timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 334–353. Springer Berlin / Heidelberg, 2005.

H. Asmuni, E. K. Burke, J. M. Garibaldi, and B. McCollum. A novel fuzzy approach to evaluate the quality of examination timetabling. In E. Burke and H. RudovÃ¡, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 327–346. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3.

J. Aubin and J. Ferland. A large scale timetabling problem. *Computers & Operations Research*, 16(1):67 – 77, 1989. ISSN 0305-0548.

P. Avella and I. Vasil'Ev. A computational study of a cutting plane algorithm for university course timetabling. *Journal of Scheduling*, 8:497–514, 2005. ISSN 1094-6136.

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

A. Azlan and N. M. Hussin. Implementing graph coloring heuristic in construction phase of curriculum-based course timetabling problem. In *Computers Informatics (ISCI), 2013 IEEE Symposium on*, pages 25–29, 2013. doi: 10.1109/ISCI.2013.6612369.

N. Balakrishnan and R. Wong. A network model for the rotating workforce scheduling problem. *Networks*, 20:25–42, 1990.

D. Banks, P. Beek, and A. Meisels. A heuristic incremental modeling approach to course timetabling. In R. Mercer and E. Neufeld, editors, *Advances in Artificial Intelligence*, volume 1418 of *Lecture Notes in Computer Science*, pages 16–29. Springer Berlin Heidelberg, 1998. ISBN 978-3-540-64575-7.

V. Bardadym. Computer-aided school and university timetabling: The new wave. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 22–45. Springer Berlin / Heidelberg, 1996.

G. Beligiannis, C. Moschopoulos, G. Kaperonis, and S. Likothanassis. Applying evolutionary computation to the school timetabling problem: The greek case. *Computers & Operations Research*, 35(4):1265 – 1280, 2008. ISSN 0305-0548.

G. Beligiannis, C. Moschopoulos, and S. Likothanassis. A genetic algorithm approach to school timetabling. *Journal of the Operatio*, 60:23–42, 2009.

G. Bello, M. Rangel, and M. Boeres. An approach for the class/teacher timetabling problem using graph coloring. In *In the proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*. PATAT2008, 2008.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

N. Boland, B. Hughes, L. Merlot, and P. Stuckey. New integer linear programming approaches for course timetabling. *Computers & Operations Research*, 35(7):2209 – 2233, 2008. ISSN 0305-0548. Part Special Issue: Includes selected papers presented at the ECCO'04 European Conference on combinatorial Optimization.

A. Bonutti, F. De Cesco, L. Di Gaspero, and A. Schaerf. Benchmarking curriculum-based course timetabling: formulations, data formats, instances, validation, visualization, and results. *Annals of Operations Research*, 194(1):59–70, April 2012. ISSN 0254-5330.

M. Bufé, T. Fischer, H. Gubbels, C. Häcker, O. Hasprich, C. Scheibel, K. Weicker, N. Weicker, M. Wenig, and C. Wolfangel. Automated solution of a highly constrained school timetabling problem - preliminary results. In E. Boers, editor, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 431–440. Springer Berlin / Heidelberg, 2001.

E. Burke and J. Newall. A multistage evolutionary algorithm for the timetable problem. *Evolutionary Computation, IEEE Transactions on*, 3(1):63 –74, apr 1999. ISSN 1089-778X.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217.

E. Burke, J. Newall, and R. Weare. A memetic algorithm for university exam timetabling. In *Selected papers from the First International Conference on Practice and Theory of Automated Timetabling*, pages 241–250, London, UK, 1996. Springer-Verlag. ISBN 3-540-61794-9.

E. Burke, Y. Bykov, J. Newall, and S. Petrovic. A time-predefined local search approach to exam timetabling problems. *IIE Transactions*, 36(6):509–528, 2004a.

E. Burke, P. De Causmaecker, G. Berghe, and H. Van Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7:441–499, 2004b. ISSN 1094-6136. 10.1023/B:JOSH.0000046076.75950.0b.

E. Burke, J. Kingston, and D. Werra. Application to timetabling. In J. Gross and J. Yellen, editors, *The Handbook of Graph Theory*, chapter 5, pages 445–474. Chapman Hall/CRC Press, 2004c.

E. Burke, M. Dror, S. Petrovic, and R. Qu. Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. In R. Sharda, S. Voß, B. Golden, S. Raghavan, and E. Wasil, editors, *The Next Wave in Computing, Optimization, and Decision Technologies*, volume 29 of *Operations Research/Computer Science Interfaces Series*, pages 79–91. Springer US, 2005. ISBN 978-0-387-23529-5.

E. Burke, B.McCollum, A. Meisels, S. Petrovic, and R. Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177 – 192, 2007. ISSN 0377-2217.

E. Burke, G. Kendall, M. Mısır, and E. Özcan. Monte carlo hyper-heuristics for examination timetabling. *Annals of Operations Research*, pages 1–18, 2010a. ISSN 0254-5330.

E. Burke, J. Marecek, A. Parkes, and H. Rudová. Decomposition, reformulation, and diving in university course timetabling. *Computers & Operations Research*, 37(3):582–597, 2010b.

E. K. Burke, J. Marec, A. J. Parkes, and H. Rudova. A branch-and-cut procedure for the udine course timetabling problem. *Annals of Operations Research*, 194(1):71–87, 2012. ISSN 0254-5330.

V. Cacchiani, A. Caprara, R. Roberti, and P. Toth. A new lower bound for curriculum-based course timetabling. *Computers & Operations Research*, 40(10):2466 – 2477, 2013. ISSN 0305-0548.

H. Cambazard, E. Hebrard, B. O'Sullivan, and A. Papadopoulos. Local search and constraint programming for the post enrolment-based course timetabling problem. In *PATAT2008: Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling*, 2008.

M. Caramia, P. Dell'Olmo, and G. Italiano. Novel local-search-based approaches to university examination timetabling. *INFORMS JOURNAL ON COMPUTING*, 20(1):86–99, 2008.

M. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193 – 202, 1986. ISSN 0030364X.

M. Carter and G. Laporte. Recent developments in practical examination timetabling. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 1–21. Springer Berlin / Heidelberg, 1996.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

M. Carter, G. Laporte, and S. Lee. Examination timetabling: Algorithmic strategies and applications. *The Journal of the Operational Research Society*, 47(3):373–383, March 1996.

M. W. Carter. A comprehensive course timetabling and student scheduling system at the university of waterloo. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 64–82. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42421-5.

S. Casey and J. Thompson. Grasping the examination scheduling problem. In E. Burke and P. De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 232–244. Springer Berlin / Heidelberg, 2003.

S. Ceschia, L. D. Gaspero, and A. Schaerf. Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. *Computers & Operations Research*, 39(7):1615 – 1624, 2012. ISSN 0305-0548.

B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447 – 460, 2003. ISSN 0377-2217.

R.-M. Chen and H.-F. Shih. Solving university course timetabling problems using constriction particle swarm optimization with local search. *Algorithms*, 6(2):227–244, 2013. ISSN 1999-4893.

M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9:403–432, 2006. ISSN 1094-6136.

A. Colorni, M. Dorigo, and V. Maniezzo. Metaheuristics for high school timetabling. *Comput. Optim. Appl.*, 9:275–298, March 1998. ISSN 0926-6003.

D. Costa and A. Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48 (3):295–305, 1997.

S. Daskalaki and T. Birbas. Efficient solutions for a university timetabling problem through integer programming. *European Journal of Operational Research*, 160(1):106 – 120, 2005. ISSN 0377-2217.

S. Daskalaki, T. Birbas, and E. Housos. An integer programming formulation for a case study in university timetabling. *European Journal of Operational Research*, 153:117–135, 2004.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2): 151 – 162, 1985. ISSN 0377-2217.

P. Demeester, B. Bilgin, P. Causmaecker, and G. Berghe. A hyperheuristic approach to examination timetabling problems: benchmarks and a new problem from practice. *Journal of Scheduling*, 15(1):83–103, 2012. ISSN 1094-6136.

M. Dimopoulou and P. Miliotis. Implementation of a university course and examination timetabling system. *European Journal of Operational Research*, 130(1):202 – 213, 2001. ISSN 0377-2217.

K. A. Dowsland and J. M. Thompson. Ant colony optimization for the examination scheduling problem. *Journal of the Operational Research Society*, 56:456–438, 2005. doi: doi:10.1057/palgrave.jors.2601830.

T.-A. Duong and K.-H. Lam. Combining constraint programming and simulated annealing on university exam timetabling, 2004.

K. Easton, G. Nemhauser, and M. Trick. Solving the travelling tournament problem: A combined integer programming and constraint programming approach. In E. Burke and P. De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 100–109. Springer Berlin / Heidelberg, 2003.

M. Eley. Ant algorithms for the exam timetabling problem. In E. K. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 364–382. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3.

ExamBenchmarks. Benchmark data sets in exam timetabling. `http://www.cs.nott.ac.uk/~rxq/data.htm`[Retrieved 28/11-2013], 2013.

C. Fernandes, J. Caldeira, F. Melicio, and A. Rosa. High school weekly timetabling by evolutionary algorithms. In *Proceedings of the 1999 ACM symposium on Applied computing*, SAC '99, pages 344–350, New York, NY, USA, 1999. ACM. ISBN 1-58113-086-4.

G. Filho and L. Lorena. A constructive evolutionary approach to school timetabling. In E. Boers, editor, *Applications of Evolutionary Computing*, volume 2037 of *Lecture Notes in Computer Science*, pages 130–139. Springer Berlin / Heidelberg, 2001.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

G. Fonseca, H. Santos, and T. Toffolo. Late acceptance-hill climbing applied to the high school timetabling problem,. In *Multidisciplinary International Scheduling Conference VI*, 2013.

L. Gaspero and A. Schaerf. Multi-neighbourhood local search with application to course timetabling. In E. Burke and P. Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 262–275. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-40699-0.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

C. Gogos, P. Alefragis, and E. Housos. An improved multi-staged algorithmic process for the solution of the examination timetabling problem. *Annals of Operations Research*, 194(1):203–221, 2012. ISSN 0254-5330.

M. Hansen and R. Vidal. Planning of high school examinations in denmark. *European Journal of Operational Research*, 87(3):519 – 534, 1995. ISSN 0377-2217. doi: DOI:10.1016/0377-2217(95)00227-8. Operational Research in Europe.

J.-K. Hao and U. Benlic. Lower bounds for the itc-2007 curriculum-based course timetabling problem. *European Journal of Operational Research*, 212(3):464 – 472, 2011. ISSN 0377-2217.

F. Jacobsen, A. Bortfeldt, and H. Gehring. Timetabling at german secondary schools: Tabu search versus constraint programming. In E. K. Burke and H. Rudova, editors, *Proceedings of the International Conference on the Practice and Theory of Automated Timetabling (PATAT 2006)*, pages 439–442, 2006. ISBN 80-210-3726-1.

W. Junginger. Timetabling in germany–a survey. *Interfaces*, 16(4):66–74, 1986. doi: 10.1287/inte.16.4.66.

M. Kahar and G. Kendall. The examination timetabling problem at universiti malaysia pahang: Comparison of a constructive heuristic with an existing software solution. *European Journal of Operational Research*, 207(2):557 – 565, 2010. ISSN 0377-2217.

G. Kendall, S. Knust, C. Ribeiro, and S. Urrutia. Scheduling in sports: An annotated bibliography. *Computers & Operations Research*, 37(1):1 – 19, 2010. ISSN 0305-0548.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

J. Kingston. A tiling algorithm for high school timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 208–225. Springer Berlin / Heidelberg, 2005.

J. H. Kingston. Educational timetabling. In A. S. Uyar, E. Ozcan, and N. Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013a. ISBN 978-3-642-39303-7.

J. H. Kingston. The hseval high school timetable evaluator. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi` [Retrieved 28/11-2013], Aug. 2013b.

P. Kostuch. Timetabling competition - sa-based heuristic. In *PATAT 2004: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, 2004.

S. Kristiansen and T. R. Stidsen. Elective course student sectioning at danish high schools. *Annals of Operations Research*, PATAT 2012 SI:To appear, 2013.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

S. Kristiansen, A. Mason, and T. R. Stidsen. High school student sectioning at danish high schools. *European Journal of Operational Research*, MISTA 2013:Submitted, 2013a.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013b.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Integer programming for the generalized (high) school timetabling problem. *Journal of Scheduling*, Submitted 5/9-2013, 2013c.

Z. Lü, J.-K. Hao, and F. Glover. Neighborhood analysis: a case study on curriculum-based course timetabling. *Journal of Heuristics*, 17(2):97–118, 2011. ISSN 1381-1231.

G. Lach and M. Lübbecke. Optimal university course timetables and the partial transversal polytope. In C. McGeoch, editor, *Experimental Algorithms*, volume 5038 of *Lecture Notes in Computer Science*, pages 235–248. Springer Berlin / Heidelberg, 2008.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

R. Lewis and B. Paechter. Application of the grouping genetic algorithm to university course timetabling. In G. R. Raidl and J. Gottlieb, editors, *Evolutionary Computation in Combinatorial Optimization*, volume 3448 of *Lecture Notes in Computer Science*, pages 144–153. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-25337-2.

R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Accounting and Finance Working Papers A2007/3, Cardiff University, Cardiff Business School, Accounting and Finance Section, 2007.

M. Lohnertz. A timetabling system for the german gymnasium. In *Proceedings of the fourth international conference on the practice and theory of automated timetabling.*, 2002.

N. Mansour, V. Isahakian, and I. Ghalayini. Scatter search technique for exam timetabling. *Applied Intelligence*, 34(2):299–310, 2011. ISSN 0924-669X.

C. H. Martin. Ohio university's college of business uses integer programming to schedule classes. *Interfaces*, 34(6):460–465, November 2004.

B. McCollum. University timetabling: Bridging the gap between research and practice. In *in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 15–35. Springer, 2006.

B. McCollum. International timetabling competition 2007. `http://www.cs.qub.ac.uk/itc2007/index.htm`[Retrieved 28/11-2013], 2007.

B. McCollum, A. Schaerf, B. Paechter, P. McMullan, R. Lewis, A. J. Parkes, L. D. Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

B. McCollum, P. McMullan, A. Parkes, E. Burke, and R. Qu. A new model for automated examination timetabling. *Annals of Operations Research*, 194:291–315, 2012. ISSN 0254-5330. 10.1007/s10479-011-0997-x.

A. Meisels and A. Schaerf. Modelling and solving employee timetabling problems. *Annals of Mathematics and Artificial Intelligence*, 39(1-2):41–59, 2003. ISSN 1012-2443.

F. Melício, P. Caldeira, and A. Rosa. Solving real school timetabling problems with meta-heuristics. In *Proceedings of the 4th WSEAS International Conference on Applied Mathematics and Computer Science*, pages 4:1–4:8, Stevens Point, Wisconsin, USA, 2005. World Scientific and Engineering Academy and Society (WSEAS). ISBN 960-8457-17-3.

F. Melício, J. P. Calderia, and A. Rosa. Thor: A tool for school timetabling. In E. K. Burke and H. Rudova, editors, *Proceedings of the 6th International Conference on the Practice and Teaching of Automated Timetabling (PATAT 2006)*, pages 532–535, 2006. ISBN 80-210-3726-1.

L. Merlot, N. Boland, and P. Hughes, B.and Stuckey. A hybrid algorithm for the examination timetabling problem. In E. Burke and P. De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 207–231. Springer Berlin / Heidelberg, 2003.

K. Minh, N. Thanh, K. Trang, and N. Hue. Using tabu search for solving a high school timetabling problem. In N. Nguyen, R. Katarzyniak, and S.-M. Chen, editors, *Advances in Intelligent Information and Database Systems*, volume 283 of *Studies in Computational Intelligence*, pages 305–313. Springer Berlin / Heidelberg, 2010.

MISTA. Multidisciplinary international scheduling conference: Theory & application. `http://www.schedulingconference.org/`[Retrieved 28/11-2013], 2013.

T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172: 429–446, 2009. ISSN 0254-5330.

T. Müller. Real-life examination timetabling. In *Proceedings of the tenth Multidisciplinary International Scheduling Conference (MISTA2013)*, 2013.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

T. Müller and H. Rudová. Real-life curriculum-based timetabling. In *Proceedings of the 9th International Conference on the Practice and Theory of Automated Timetabling, PATAT2012*, 2012.

T. Müller, H. Rudová, and R. Barták. Minimal perturbation problem in course timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 126–146. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

C. N. Moschopoulos, C. E. Alexakos, C. Dosi, G. N. Beligiannis, and S. D. Likothanassis. A user-friendly evolutionary tool for high-school timetabling. In C. Koutsojannis and S. Sirmakessis, editors, *Tools and Applications with Artificial Intelligence*, volume 166 of *Studies in Computational Intelligence*, pages 149–162. Springer Berlin Heidelberg, 2009. ISBN 978-3-540-88068-4.

A. Moura and R. Scaraficci. A grasp strategy for a more constrained school timetabling problem. *International Journal of Operational Research*, 7:152–170(19), 2010.

K. Murray, T. Müller, and H. Rudová. Modeling and solution of a complex university course timetabling problem. In E. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 189–209. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-77344-3.

K. Nguyen, Q. Nguyen, H. Tran, P. Nguyen, and N. Tran. Variable neighborhood search for a real-world curriculum-based university timetabling problem. In *Knowledge and Systems Engineering (KSE), 2011 Third International Conference on*, pages 157 –162, oct. 2011.

C. Nothegger, A. Mayer, A. Chwatal, and G. R. Raidl. Solving the post enrolment course timetabling problem by ant colony optimization. *Annals of Operations Research*, 194(1):325–339, 2012. ISSN 0254-5330.

K. Nurmi and J. Kyngas. A framework for school timetabling problem. In *Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications*, pages 386–393, 2007.

K. Nurmi and J. Kyngas. A conversion scheme for turning a curriculum-based timetabling problem into a school timetabling problem. In *Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling (PATAT 2008)*, 2008.

E. Ozcan, Y. Bykov, M. Birben, and E. Burke. Examination timetabling using late acceptance hyper-heuristics. In *Evolutionary Computation, 2009. CEC '09. IEEE Congress on*, pages 997 –1004, may 2009.

B. Paechter, L. M. Gambardella, and O. Rossi-Doria. International timetabling competition 2003. `http://www.idsia.ch/Files/ttcomp2002/oldindex.html`[Retrieved 28/11-2013], 2002.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3):230–238, 2003.

PATAT. International conference on the practice and theory of automated timetabling. `http://www.patatconference.org/`[Retrieved 28/11-2013], 2013.

S. Petrovic and Y. Bykov. A multiobjective optimisation technique for exam timetabling based on trajectories. In E. Burke and P. De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, volume 2740 of *Lecture Notes in Computer Science*, pages 181–194. Springer Berlin / Heidelberg, 2003.

N. Pillay. A survey of school timetabling research. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

N. Pillay and W. Banzhaf. A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem. *European Journal of Operational Research*, 197(2):482 – 491, 2009. ISSN 0377-2217.

M. Pimmer and G. R. Raidl. A timeslot-filling heuristic approach to construct high-school timetables. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Advances in Metaheuristics*, volume 53 of *Operations Research/Computer Science Interfaces Series*, pages 143–157. Springer New York, 2013. ISBN 978-1-4614-6321-4.

G. Post. Benchmarking project for (high) school timetabling. http://www.utwente.nl/ctit/hstt/ [Retrieved 28/11-2013], Aug. 2013.

G. Post, J. Kingston, A. Schaerf, L. D. Gaspero, and B. McCollum. International timetabling competition 2011. http://www.utwente.nl/ctit/hstt/itc2011/[Retrieved 28/11-2013], 2011.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194:385–397, 2012a. ISSN 0254-5330.

G. Post, S. Ahmadi, and F. Geertsema. Cyclic transfers in school timetabling. *OR Spectrum*, 34 (1):133–154, 2012b. ISSN 0171-6468.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012c.

G. Post, L. Gaspero, J. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. *Annals of Operations Research*, February 2013. ISSN 0254-5330.

G. F. Post and H. W. A. Ruizenaar. Clusterschemes in dutch secondary schools. Memorandum 1707, Department of Applied Mathematics, University of Twente, Enschede, 2004.

PurdueBenchmark. The purdue benchmark data. http://www.unitime.org/uct_datasets.php[Retrieved 28/11-2013], 2013.

PurdueExamBenchmark. The purdue benchmark data. http://www.unitime.org/exam_datasets.php[Retrieved 28/11-2013], 2013.

R. Qu and E. K. Burke. Adaptive decomposition and construction for examination timetabling problems. In *Proceedings of the 3rd multidisciplinary international scheduling conference: theory and applications*, 2007.

R. Qu, E. Burke, B. McCollum, L. Merlot, and S. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89, 2009. ISSN 1094-6136.

A. Qualizza and P. Serafini. A column generation scheme for faculty timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 161–173. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

R. Raghavjee and N. Pillay. Evolving solutions to the school timetabling problem. In *Nature Biologically Inspired Computing, 2009. NaBIC 2009. World Congress on*, pages 1524–1527, 2009.

R. Raghavjee and N. Pillay. An informed genetic algorithm for the high school timetabling problem. In *Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists*, SAICSIT '10, pages 408–412, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-950-3.

S. A. Rahman, A. Bargiela, E. K. Burke, E. Özcan, B. McCollum, and P. McMullan. Adaptive linear combination of heuristic orderings in constructing examination timetables. *European Journal of Operational Research*, 232(2):287 – 297, 2014. ISSN 0377-2217.

S. Ribic and S. Konjicija. A two phase integer linear programming approach to solving the school timetable problem. In *Information Technology Interfaces (ITI), 2010 32nd International Conference on*, pages 651–656, 2010.

V. Robert and A. Hertz. How to decompose constrained course scheduling problems into easier assignment type subproblems. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 364–373. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-61794-5.

J. Romrös and J. Homberger. An evolutionary algorithm for high school timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 485–488. SINTEF, 2012.

H. Rudova and K. Murray. University course timetabling with soft constraints. In *Practice And Theory of Automated Timetabling IV.*, pages 310–328, 2003.

H. Rudová, T. Müller, and K. Murray. Complex university course timetabling. *Journal of Scheduling*, 14(2):187–207, 2011. ISSN 1094-6136.

N. R. Sabar, M. Ayob, and G. Kendall. Solving examination timetabling problems using honeybee mating optimization (etp-hbmo). In *Proceedings of the 4rd multidisciplinary international scheduling conference: theory and applications*, 2009.

N. R. Sabar, M. Ayob, R. Qu, and G. Kendall. A graph coloring constructive hyper-heuristic for examination timetabling problems. *Applied Intelligence*, 37(1):1–11, 2012. ISSN 0924-669X.

S. E. Sampson, J. R. Freeland, and E. N. Weiss. Class scheduling to maximize participant satisfaction. *Interfaces*, 25(3):30–41, 1995.

H. Santos, L. Ochi, and M. Souza. An efficient tabu search heuristic for the school timetabling problem. In C. Ribeiro and S. Martins, editors, *Experimental and Efficient Algorithms*, volume 3059 of *Lecture Notes in Computer Science*, pages 468–481. Springer Berlin / Heidelberg, 2004.

H. Santos, L. Ochi, and M. Souza. A tabu search heuristic with efficient diversification strategies for the class/teacher timetabling problem. *J. Exp. Algorithmics*, 10, December 2005. ISSN 1084-6654.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999a. ISSN 0269-2821.

A. Schaerf. Local search techniques for large high school timetabling problems. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 29(4):368 –377, jul 1999b. ISSN 1083-4427.

K. Schimmelpfeng and S. Helber. Application of a real-world university-course timetabling model solved by integer programming. *OR Spectrum*, 29:783–803, 2007. ISSN 0171-6468.

G. Schmidt and T. Ströhlein. Timetable construction – an annotated bibliography. *The Computer Journal*, 23(4):307–316, 1980.

D.-F. Shiau. A hybrid particle swarm optimization for a university course scheduling problem with flexible preferences. *Expert Systems with Applications*, 38(1):235 – 248, 2011. ISSN 0957-4174.

T. Sönmez and M. U. Ünver. Course bidding at business schools*. *International Economic Review*, 51(1):99–123, 2010. ISSN 1468-2354.

K. Socha, M. Sampels, and M. Manfrin. Ant algorithms for the university course timetabling problem with regard to the state-of-the-art. In S. Cagnoni, C. G. Johnson, J. J. R. Cardalda, E. Marchiori, D. W. Corne, J.-A. Meyer, J. Gottlieb, M. Middendorf, A. Guillot, G. Raidl, and E. Hart, editors, *Applications of Evolutionary Computing*, volume 2611 of *Lecture Notes in Computer Science*, pages 334–345. Springer Berlin Heidelberg, 2003. ISBN 978-3-540-00976-4.

M. Souza, N. Maculan, and L. Ochi. A grasp-tabu search algorithm for school timetabling problems. In M. Resende and J. de Sousa, editors, *Metaheuristics: Computer decision-making*, pages 659–672. Kluwer Academic Publishers, 2003.

M. Sørensen and F. H. W. Dahms. A two-stage decomposition of high school timetabling applied to cases in denmark. *Computers & Operations Research*, 43:36–49, March 2014.

M. Sørensen and T. R. Stidsen. Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. *Annals of Operations Research*, PATAT 2012 SI:Submitted Jan 21. 2013, 2013.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

S. Suyanto. An informed genetic algorithm for university course and student timetabling problems. In *Proceedings of the 10th international conference on Artifical intelligence and soft computing: Part II*, ICAISC'10, pages 229–236, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13231-6, 978-3-642-13231-5.

J. Thompson and K. Dowsland. A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7-8):637 – 648, 1998. ISSN 0305-0548.

H. Turabieh and S. Abdullah. An integrated hybrid approach to the examination timetabling problem. *Omega*, 39(6):598 – 607, 2011a. ISSN 0305-0483.

H. Turabieh and S. Abdullah. A hybrid fish swarm optimisation algorithm for solving examination timetabling problems. In C. Coello, editor, *Learning and Intelligent Optimization*, volume 6683 of *Lecture Notes in Computer Science*, pages 539–551. Springer Berlin Heidelberg, 2011b. ISBN 978-3-642-25565-6.

H. Turabieh, S. Abdullah, B. McCollum, and P. McMullan. Fish swarm intelligent algorithm for the course timetabling problem. In J. Yu, S. Greco, P. Lingras, G. Wang, and A. Skowron, editors, *Rough Set and Knowledge Technology*, volume 6401 of *Lecture Notes in Computer Science*, pages 588–595. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-16247-3.

UdineBenchmark. The udine benchmark data. `http://satt.diegm.uniud.it/projects/`[Retrieved 28/11-2013], 2013.

UniTime. Unitime publications. `http://www.unitime.org/publications.php`[Retrieved 28/11-2013], 2013.

C. Valouxis and E. Housos. Constraint programming approach for school timetabling. *Computers & Operations Research*, 30(10):1555 – 1572, 2003. ISSN 0305-0548. Part Special Issue: Analytic Hierarchy Process.

C. Valouxis, C. Gogos, P. Alefragis, and E. Housos. Decomposing the high school timetable problem. In *Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

G. White and B. Xie. Examination timetables and tabu search with longer-term memory. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 85–103. Springer Berlin / Heidelberg, 2001.

G. White, B. Xie, and S. Zonjic. Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, 153(1):80 – 91, 2004. ISSN 0377-2217. Timetabling and Rostering.

P. Wilke, M. Gröbner, and N. Oster. A hybrid genetic algorithm for school timetabling. In B. McKay and J. Slaney, editors, *AI 2002: Advances in Artificial Intelligence*, volume 2557 of *Lecture Notes in Computer Science*, pages 455–464. Springer Berlin / Heidelberg, 2002.

T. Wong, P. Cote, and P. Gely. Final exam timetabling: a practical approach. In *Electrical and Computer Engineering, 2002. IEEE CCECE 2002. Canadian Conference on*, volume 2, pages 726 – 731 vol.2, 2002. doi: 10.1109/CCECE.2002.1013031.

J. Wood and D. Whitaker. Student centred school timetabling. *The Journal of the Operational Research Society*, 49(11):1146–1152, Nov. 1998.

A. Wren. Scheduling, timetabling and rostering — a special relationship? In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 46–75. Springer Berlin / Heidelberg, 1996.

Y. Yang and S. Petrovic. A novel similarity measure for heuristic selection in examination timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 247–269. Springer Berlin / Heidelberg, 2005.

T. Yigit. Constraint-based school timetabling using hybrid genetic algorithms. In R. Basili and M. Pazienza, editors, *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing*, volume 4733 of *Lecture Notes in Computer Science*, pages 848–855. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-74781-9.

E. Özcan, B. Bilgin, and E. E. Korkmaz. A comprehensive analysis of hyper-heuristics. *Intelligent Data Analysis*, 12:3–23, 2008.

D. Zhang, Y. Liu, R. M'Hallah, and S. Leung. A simulated annealing with a new neighborhood structure based algorithm for high school timetabling problems. *European Journal of Operational Research*, 203(3):550 – 558, 2010. ISSN 0377-2217.

Part II

High School Timetabling using the XHSTT format

Chapter 3

International Timetabling Competition 2011: An Adaptive Large Neighborhood Search algorithm

Matias Sørensen*† Simon Kristiansen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
msso@dtu.dk, sikr@dtu.dk, thst@dtu.dk

† MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

3.1 Introduction

An algorithm based on Adaptive Large Neighborhood Search (ALNS) for solving the generalized High School Timetabling problem in XHSTT-format (Post et al. (2012a)) is presented. This algorithm was among the finalists of round 2 of the International Timetabling Competition 2011 (ITC2011). For problem description and results we refer to Post et al. (2012b).

3.2 Adaptive Large Neighborhood Search

Adaptive Large Neighborhood Search was first developed as a metaheuristic for the class of Vehicle Routing Problems (Pisinger and Ropke (2005); Ropke and Pisinger (2006)). It has been applied for few other problem classes as well, including Project Scheduling (Muller (2009, 2010)), Lot-sizing (Muller et al. (2011)), Optimal Statistic Median Problem (Katterbauer et al. (2012)).

Recently we have developed a framework based on ALNS for solving combinatorial optimization

[1] Conference abstract published in proceedings of *PATAT 2012* (Sørensen et al. (2012))

problems (written in C# 4.0). This framework is part of the commercial product Lectio[2], where it is used to solve various practical timetabling problems, see Kristiansen et al. (2011); Sørensen and Stidsen (2012) and Kristiansen and Stidsen (2012).

The pseudo code for a general ALNS algorithm is given in Algorithm 1.

Algorithm 1: Adaptive Large Neighborhood Search

```
1  Candidate solution x, remove-methods Ω⁻, insert-methods Ω⁺
2  x_best = x
3  while stop-criterion not met do
4      x' = x
5      RemoveStrategy: select q as some quantity to be removed
6      AdaptiveStrategy: select remove-method r ∈ Ω⁻ and insert-method i ∈ Ω⁺
7      remove requests from x' using r(q)
8      insert requests into x' using i
9      AdaptiveStrategy: update performance indicators
10     if c(x') ≤ c(x_best) then
11         x_best = x'
12     AcceptStrategy: set candidate solution x to either x', x_best or x itself
13 return x_best
```

The main points of the algorithm are described below in general terms.

- In each iteration, a remove and insertion method is chosen and applied to the candidate solution. The combination of these methods defines the neighborhood of the algorithm, hence there exists $|\Omega^-| \cdot |\Omega^+|$ different neighborhoods.
- `RemoveStrategy`: Governs the selection of q. This has major influence on how much computational time each iteration requires.
- `AdaptiveStrategy`: Responsible for selecting remove and insertion methods in each iteration, and updating their respective performance indicators of these method by some metric.
- `AcceptStrategy`: Determines which solution to use as candidate solution for next iteration. This could in principle be any known solution, but is usually selected as either the current candidate solution x itself, the newly produced solution x', or the current best solution x_{best}.

3.3 Algorithm Setup for ITC2011

Here we describe our implementation of a ALNS algorithm for the XHSTT format. The choice of ALNS strategies are briefly mentioned below. More details will be available in the full paper.

- `RemoveStrategy`: The remove and insertion methods deal with sub-events. q is defined as the sum of the duration of the sub-events which are removed from the solution. We select q as a random number, bounded by a percentage of the total duration of all instance events.
- `AdaptiveStrategy`: We have chosen a metric essentially based on two parameters for each method; The number of times the method was part of an iteration which yielded a better solution than the current one, and the relative gap between the current solution and the resulting solution from applying the method.
- `AcceptStrategy`: An acceptance criteria borrowed from Simulated Annealing (SA) is used, with the following additional property: If no new best solution has been found in a number of iterations, the temperature is increased by a factor, and the candidate solution is set to

[2] http://www.lectio.dk
Cloud-based administration system for high schools. Developed by MaCom A/S, Vesterbrogade 48 1., 1620 Copenhagen V, Denmark

the best known solution. The intention is to allow more diverse exploring of the area around the best known solution, in case the algorithm gets 'stuck'.

Let a *move* be a small perturbation on a solution. The following moves are used in this implementation: Move $M_{se,t}$ denotes the assigning of sub-event se to time t. $M_{r,er,se}$ denotes the assigning of resource r to event resource er on sub-event se. Furthermore we also implement the corresponding unassign-moves, denoted $M^{\neg}_{se,t}$ and $M^{\neg}_{r,er,se}$, respectively.

Using these moves a total of 9 insertion methods (all more or less based on the greedy principle, e.g. regret heuristics (Potvin and Rousseau (1993); Sørensen and Stidsen (2012)), and 14 remove methods (all based on some element of relatedness and an element of randomness) are implemented. These methods are divided into three categories, based on what they (un-)assign: Only times, only resources, or both times and resources.

An example of a remove method is the following, which removes sub-events from non-preferred times: Given an XHSTT instance, and a solution S to this instance. Find all tuples $\langle se, t\rangle$ of S, where sub-event se is assigned time t, and t is not a preferred time for sub-event se (see *Prefer times constraints*, Kingston (2010)). Let the set of these tuples be denoted U. Select randomly a subset of these tuples $\overline{U} \subseteq U$ such that the sum of the duration of all sub-events of the tuples in $\overline{U}$ equals q. Perform an unassign time move $M^{\neg}_{se,t}$ for each of the tuples in $\overline{U}$.

An example of an insertion method is the following: Let $\Delta(M) \in \mathbb{R}$ be the profit of performing move M on the solution at hand S. Select $M_{\text{best}} = \arg\min_{se,t}(\Delta(M_{se,t}))$, and if $\Delta(M_{\text{best}}) \leq 0$, apply M_{best} to S and repeat, otherwise stop. This is a greedy method which assigns times to sub-events, until no profitable move can be found.
In the full paper all insert/remove methods will be described in detail.

The final algorithm contains 9 free parameters, which were tuned for best performance using the `irace` package (see López-Ibáñez et al. (2011); Birattari (2005)).

3.4 Final Remarks

This paper documents how Adaptive Large Neighborhood Search can be applied to problems in XHSTT format.

The proposed algorithm was applied to all instances in archive XHSTT-ITC2011, and showed competitive results in most cases (comparing to the best known solutions at that point in time).

ALNS has not been used much in the field of timetabling, but we see no reason to believe that ALNS should not perform well on other (related) problems in this field.

Acknowledgements Thank you goes to Michael Herold for fruitful discussions concerning ALNS strategies. Thank you to Manuel López-Ibáñez for help using the `irace` package. And finally thank you to David Pisinger for advice on ALNS implementation.

Bibliography

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

K. Katterbauer, C. Oguz, and S. Salman. Hybrid adaptive large neighborhood search for the optimal statistic median problem. *Computers & Operations Research*, 39(11):2679 – 2687, 2012. ISSN 0305-0548.

J. H. Kingston. The hseval high school timetable evaluator, 2010. URL `http://it.usyd.edu.au/~jeff/hseval.cgi`.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

M. López-Ibáñez, J. Dubois-Lacoste, T. Stützle, and M. Birattari. The irace package: Iterated racing for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, Université Libre de Bruxelles, IRIDIA, Av F. D. Roosevelt 50, CP 194/6 1050 Bruxelles, Belgium, Februrary 2011. http://iridia.ulb.ac.be/irace.

L. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

L. Muller. An adaptive large neighborhood search algorithm for the multi-mode resource-constrained project scheduling problem. l, Department of Management Engineering, Technical University of Denmark Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark, 2010.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194:385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

M. Sørensen and T. R. Stidsen. High school timetabling: Modeling and solving a large number of cases in denmark. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 359–364. SINTEF, 2012.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

Chapter 4

Integer Programming for the Generalized (High) School Timetabling Problem

Simon Kristiansen*† Matias Sørensen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, msso@dtu.dk, thst@dtu.dk

†MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

Abstract Recently the XHSTT format for (High) School Timetabling was introduced, which provides a uniform way of modeling problem instances and corresponding solutions. The format supports a big variety of constraints, and currently 38 real-life instances from 11 different countries are available. Thereby the XHSTT format serves as a common ground for researchers within this area. This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a Mixed-Integer linear Programming (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results show that our approach is able to find previously unknown optimal solutions for 2 instances of XHSTT, and proves optimality of 4 known solutions. For the instances not solved to optimality, new non-trivial lower bounds were found in 11 cases, and new best-known solutions were found in 9 cases. Furthermore the approach is shown to be competitive with the finalist of Round 2 of the International Timetabling Competition 2011.

4.1 Introduction

The problem of scheduling lectures to time slots and/or resources at high schools is known as the *High School Timetabling* (HST) problem. This is an important problem for high schools in many countries, and a large amount of different solution approaches have been proposed, see the survey Schaerf (1999).

It is well recognized that the specifications of the HST problem varies significantly depending on the country of which the problem originates, and that the problem in general is hard to solve. With the introduction of the XHSTT format (Post et al., 2012a), a large number of instances

[1]Submitted and under revision at *Journal of Scheduling (2013)*

from various origins became publicly available in standardized form. The format is based on the *Extensible Markup Language* (XML) standard, and all instances are available online (Post, 2013b). One purpose of the format is to serve as a common test-bed for *school timetabling*, in an attempt to promote research within this area. In this context, "school timetabling" denotes the area covering high school timetabling and *university course timetabling*, as the format has also been shown capable of modeling some instances of the latter problem (see Kingston (2013a) for an overview of educational timetabling problems).

This paper describes the first exact method capable of handling an arbitrary instance of the XHSTT format. The method is based on a *Mixed-Integer linear Programming* (MIP) model, which is solved in two steps with a commercial general-purpose MIP solver. Computational results are performed for all the real-life instances currently available. Thereby we are able to find previously unknown optimal solutions, and prove optimality of already known solutions.

To the best of our knowledge, all previous solution methods for the XHSTT format have been heuristic in nature. Therefore no proof of optimality has been made for any instance, except for those instances where a solution with objective value 0 is known, since 0 is a trivial lower bound for any XHSTT instance. The obvious advantage of *Integer Programming* (IP) over heuristic methods is the capability to issue certificates of optimality. Therefore it is remarked that a big advance within general-purpose MIP solvers has happened in recent years, see e.g. Bixby (2012). Even though the MIP we will present is inevitable complex in nature, it will be shown that it can be used to find optimal solutions for several instances of the XHSTT archive `ALL_INSTANCES`. For those instances where an optimal solution cannot be found, we are able to show a non-trivial lower bound on optimum in the majority of cases. These are significant results for high school timetabling in general.

The outline of this paper is as follows. Section 4.2 presents related literature. Section 4.3 presents the MIP model of XHSTT. Section 4.4 describes computational results. Finally, Section 4.5 concludes and describes future research possibilities.

4.2 Related Literature

The Third International Timetabling Competition (ITC2011) considered the HST Problem, based on instances of the XHSTT format (Post et al., 2012b). Four teams were part of the final round: The overall winner (Team Goal) used Simulated Annealing and Iterated Local Search to perform local search around a generated initial solution (Fonseca et al., 2012). Participant from the University of Nottingham (HySTT) used a method based on Hyper-heuristics (Kheiri et al., 2012). Team Lectio used Adaptive Large Neighborhood Search (ALNS) (Sørensen et al., 2012). Romrös and Homberger (2012) (Team HFT) used an Evolutionary Algorithm. The results of the competition can be found at the official homepage of ITC2011 (Post (2013a)).

Pimmer and Raidl (2013) describe a 'timeslot-filling' heuristic for XHSTT, which iteratively fills selected timeslots with sets of events. Two state-of-the-art solutions were found for instances of the archive `XHSTT-2012`. Ter Braak (2012) presents a Hyper-heuristic and several other heuristics for the XHSTT.

Valouxis et al. (2012) describe a two-phase approach based on MIP used to solve the Greek case of the HST problem. This includes two instances which are part of the XHSTT project, and which were both solved to optimality (solutions were found with an objective value of 0).

In terms of Integer Programming and HST problems not based on XHSTT, the following contribution are mentioned: Santos et al. (2012) present a *Column Generation* approach for establishing bounds for a set of datasets originating from Brazil. Birbas et al. (2009) present an approach for Greek datasets where the *Shift Assignment Problem* is solved first, and the timetable is constructed on the basis on these work-shifts for teachers. The paper of Sørensen and Stidsen (2013) describes a complex MIP of the Danish case of high school timetabling, and establishes computational results for 100 real-life instances. Avella et al. (2007) present an algorithm based on Very Large-Scale Neighborhood search where the neighborhood is explored by a MIP, for Italian cases of high school timetabling.

Table 4.1: Different constraint types in the XHSTT format (Post et al., 2012b)

Constraint	Description
Assign Resource	Event resource should be assigned a resource
Assign Time	Event should be assigned a time
Split Events	Event should split into a constrained number of sub-events
Distribute Split Events	Event should split into sub-events of constrained durations
Prefer Resources	Event resource assignment should come from resource group
Prefer Times	Event time assignment should come from time group
Avoid Split Assignments	Set of event resources should be assigned the same resource
Spread Events	Set of events should be spread evenly through the cycle
Link Events	Set of events should be assigned the same time
Order Events	Set of events should be ordered
Avoid Clashes	Resource's timetable should not have clashes
Avoid Unavailable Times	Resource should not be busy at unavailable times
Limit Idle Times	Resource's timetable should not have idle times
Cluster Busy Times	Resource should be busy on a limited number of days
Limit Busy Times	Resource should be busy a limited number of times each day
Limit Workload	Resource's total workload should be limited

4.3 Problem Description and a Mixed Integer Programming Formulation

In this section a brief description of the specifications of the XHSTT format is given, and a MIP model is formulated. The entire documentation of XHSTT is available at Post (2013b). We do not intend to describe all properties of the format, but only those necessary to formulate the MIP.

An instance of XHSTT consists of *times* (denoted $\mathcal{T}$ in the following), *time groups* (denoted $\mathcal{TG}$), *resources* (denoted $\mathcal{R}$), *events* (denoted $\mathcal{E}$), *event groups* (denoted $\mathcal{EG}$) and *constraints* (denoted $\mathcal{C}$). An event $e \in \mathcal{E}$ has a duration $D_e \in \mathbb{N}$, and a number of *event resources* which we each denote $er \in e$. An event resource defines the requirement of the assignment of a resource to the event, and this resource can be specified to be preassigned. If the resource is not preassigned, a resource of proper *type* must be assigned. Furthermore an event resource er can undertake a specific role_{er}, which is used to link the event resource to certain constraints.

It is the job of any solver for XHSTT to decide how each event should be split into *sub-events*. A sub-event se is defined as a fragment of a specific event $e \in \mathcal{E}$, has a duration $D_{se} \leq D_e$, and inherits the requirement of resources defined by the event, such that each sub-event has the exact same resource requirements as the event. Let $\mathcal{SE}$ denote the entire set of sub-events, and let $se \in e$ specify that sub-event se is part of event e. The total duration of all sub-events for event $e \in \mathcal{E}$ in a solution cannot exceed D_e. In our model formulation we create the 'full set' of sub-events with different lengths, i.e. all possible combinations of sub-events for a given event can be handled. E.g. if an event has duration 4, the set of sub-events for this event has the respective lengths $1, 1, 1, 1, 2, 2, 3$ and 4. As a constraint it is then specified that the summed duration of the *active* sub-events in a solution must equals 4. A sub-event is active if it is assigned a starting time or a non-preassigned resource. An active sub-event is analogous to the concept of *solution events* defined in the XHSTT documentation.

The times $\mathcal{T}$ are ordered in chronological order, and we let $\rho(t)$ denote the index number of time t in $\mathcal{T}$. A time group $\mathcal{TG}$ defines a set of times, and we let $t \in tg$ denote that time t is part of time group tg.

Each constraint $c \in \mathcal{C}$ is of a specific type, and the set $\mathcal{C}$ can contain several constraints of the same type. Each constraint applies to certain events, event groups or resources, and penalizes certain characteristics of the timetable for these entities.

The following notation shorthand is made: By the notions $e \in c$, $r \in c$, $eg \in c$ we denote that

constraint $c \in \mathcal{C}$ applies to event $e \in \mathcal{E}$, resource $r \in \mathcal{R}$, and event group $eg \in \mathcal{EG}$, respectively.

The set of resources and times are both extended with a dummy-index, denoted the *dummy-resource* r_D and the *dummy-time* t_D, respectively. These are necessary to ensure feasibility as we create all combinations of sub-events for each event, and not all of these can be assigned a time or the required resources without the duration of the active sub-events exceeding the duration of the event. Thereby these dummy-elements in fact represent that an event resource is *not* assigned a resource, and that a sub-event is *not* assigned a starting time, respectively.

4.3.1 Objective Function

Each XHSTT constraint penalizes timetables with certain characteristics, which contributes to the objective function of the MIP. Each constraint $c \in \mathcal{C}$ has a set of *point-of-applications* (indexed by $p \in c$). With each point-of-application is associated a set of *deviations* (indexed by $d \in p$), and each deviation has a non-negative cost associated with it. How this cost is calculated depends on the constraint type. The cost of a point-of-application is found on basis of the cost of the deviations, and is influenced by an indication on the constraint whether the constraint is a *hard* or a *soft* constraints, the *weight* of the constraint ($\omega_c \in \mathbb{N}$) and an indication of which *CostFunction* to use. For each constraint $c \in \mathcal{C}$ we let the variable $s_{c,p,d} \in \mathbb{N}$ be the penalty value of the deviation $d \in p$ of the point-of-application $p \in c$. The set of point-of-applications and deviations should be understood in an abstract context; E.g. depending on the type of the constraint, a point-of-application could be an event, a resource, etc., and likewise for the deviations.

The objective of a solution consists of a value for both the hard constraints (denoted *hard cost*) and a value for the soft constraints (denoted *soft cost*). Usually the objective value of a solution is written as (hard cost, soft cost). The hard cost always takes priority over the soft cost, i.e. solutions are first ranked on their hard cost, and secondly on the soft cost. How this type of objective function is handled in context of a MIP is described in Section 4.3.4.

The cost of a constraint $c \in \mathcal{C}$ which contains slack variable $s_{c,p,d}$ is denoted $f(s_{c,p,d})$,

$$f(s_{c,p,d}) = \omega_c \cdot \text{CostFunction}(s_{c,p,d}) \tag{4.3.2}$$

Five different types of CostFunction are allowed. The most trivial one is `Sum`, which simply sums the penalty value of all deviations for all point-of-applications. In the following each CostFunction is formulated in linear terms. Let the variable $\text{obj}_c \in \mathbb{N}_0$ denote the value of the of the CostFunction of constraint $c \in \mathcal{C}$.

- **Sum:** Sum the deviations.

$$\text{obj}_c = \sum_{p \in c, d \in p} s_{p,d,c} \quad \forall c \in \mathcal{C} \tag{4.3.3}$$

- **SumSquare:** Sum the squares of the deviations.
  To cope with this non-linear cost function, the variable $s_{c,p,d,i} \in \{0,1\}$ is introduced, which takes value 1 if the deviation $d \in p$ of the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has the penalty $i \in \mathcal{I}$, and 0 otherwise. The objective value is defined as follows:

$$\text{obj}_c = \sum_{p \in c, d \in p, i \in \mathcal{I}} i^2 \cdot s_{c,p,d,i} \quad \forall c \in \mathcal{C} \tag{4.3.4}$$

  However we also need to make sure that only a single integer value is selected,

$$\sum_{i \in \mathcal{I}} s_{c,p,d,i} = 1 \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{4.3.5}$$

  The amount of elements in the set $\mathcal{I}$ determines the maximum possible penalty for a deviation, and thereby influence the maximum possible penalty for a constraint. To maintain optimality of the model, it is therefore important that the size of $\mathcal{I}$ is selected sufficiently large. This is elaborated in Section 4.4.

- **SquareSum:** Square the sum of deviations.
  The binary slack variable $u^{\text{squaresum}}_{c,p,j} \in \{0,1\}$ is introduced, which takes value 1 if the point of application $p \in c$ of constraint $c \in \mathcal{C}$ has the deviation $j \in \mathcal{J}$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, j \in \mathcal{J}} j^2 \cdot u^{\text{squaresum}}_{c,p,j} \quad \forall c \in \mathcal{C} \tag{4.3.6}$$

$$\sum_{j \in \mathcal{J}} u^{\text{squaresum}}_{c,p,j} = 1 \quad \forall c \in \mathcal{C}, p \in c \tag{4.3.7}$$

$$\sum_{d \in p} s_{p,d,c} = \sum_{j \in \mathcal{J}} j \cdot u^{\text{squaresum}}_{c,p,j} \quad \forall c \in \mathcal{C}, p \in c \tag{4.3.8}$$

  Like the set $\mathcal{I}$, the size of the set $\mathcal{J}$ must be selected sufficiently large to maintain optimality, see Section 4.4.

- **SumStep:** This penalizes by the number of positive deviations, irrespective of their value.
  The binary variable $u^{\text{sumstep}}_{c,p,d} \in \{0,1\}$ is introduced, which takes value 1 iff $s_{c,p,d} > 0$ for constraint $c \in \mathcal{C}$, point-of-application $p \in c$ and deviation $d \in p$, and 0 otherwise.

$$\text{obj}_c = \sum_{p \in c, d \in p} u^{\text{sumstep}}_{c,p,d} \quad \forall c \in \mathcal{C} \tag{4.3.9}$$

$$M \cdot u^{\text{sumstep}}_{c,p,d} \geq s_{c,p,d} \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{4.3.10}$$

  where $M \in \mathbb{N}$ is some sufficiently large number.

- **StepSum:** This CostFunction penalizes by investigating whether the constrain contains at least one positive deviation. If this is not the case, the penalty is 0.
  The binary variable $u^{\text{stepsum}}_{c} \in \{0,1\}$ is introduced, which takes value 1 if there exists at least one positive deviation for the constraint $c \in \mathcal{C}$, and 0 otherwise.

$$\text{obj}_c = u^{\text{stepsum}}_{c} \quad \forall c \in \mathcal{C} \tag{4.3.11}$$

$$M \cdot u^{\text{stepsum}}_{c} \geq s_{c,p,d} \quad \forall c \in \mathcal{C}, p \in c, d \in p \tag{4.3.12}$$

  where $M \in \mathbb{N}$ is some sufficiently large number.

4.3.2 Mixed-Integer Programming Formulation

In this section the variables and the constraints of the MIP are described. As a basis for our approach is the variable $x_{se,t,er,r} \in \{0,1\}$, which takes value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time and resource $r \in er$ is assigned to event resource $er \in se$, and 0 otherwise. To simplify notation, and to reduce the amount of non-zeros in the MIP, three auxiliary variables are introduced which all 'inherits' their values directly from $x_{se,t,er,r}$. Let the binary variable $y_{se,t} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ has been assigned time $t \in \mathcal{T}$ as starting time, and 0 otherwise. The variable $v_{t,r} \in \mathbb{N}_0$ denotes the number of times resource r is used in time t by any set of sub-events. Let variable $w_{se,er,r} \in \{0,1\}$ take value 1 if sub-event $se \in \mathcal{SE}$ is assigned resource $r \in \mathcal{R}$ for event resource er, and 0 otherwise.

Base Constaints

Besides all the constraints described in the specifications of the XHSTT, some basic constraints are needed to ensure feasibility. First of all we need to make sure that a sub-event is assigned only one starting time and that the number of resource assigned is exactly the same as the number of event resources of the event.

$$\sum_{t \in \mathcal{T}, r \in er} x_{se,t,er,r} = 1 \quad \forall se \in \mathcal{SE}, er \in se \tag{4.3.13}$$

The following constrains variable $y_{se,t}$, and together with (4.3.13) ensures that a sub-event is not spread across multiple times. We denote by $|er|_{se}$ the amount of event resources for event e of sub-event se.

$$\sum_{er \in se, r \in er} x_{se,t,er,r} = |er|_{se \in e} \cdot y_{se,t} \quad \forall se \in \mathcal{SE}, t \in \mathcal{T} \tag{4.3.14}$$

The link to variable $v_{t,r}$ is shown in eq. (4.3.16). For time $t \in \mathcal{T}$ and $se \in \mathcal{SE}$ is found the set of possible starting-times for se which will cause resource $r \in \mathcal{R}$ to be used in time $t \in \mathcal{T}$. Let the set $T^{\text{start}}_{se,t} \subseteq \mathcal{T}$ be the set of times which sub-event se lies in if it is assigned starting time t, i.e.

$$T^{\text{start}}_{se,t} = \{t' \in \mathcal{T} \setminus t_D \mid \rho(t) - D_{se} + 1 \leq \rho(t') \leq \rho(t)\} \tag{4.3.15}$$

$$\sum_{se \in \mathcal{SE}, er \in se, t' \in T^{\text{start}}_{se,t}} x_{se,t',er,r} = v_{t,r} \quad \forall t \in \mathcal{T} \setminus t_D, r \in \mathcal{R} \tag{4.3.16}$$

The link to variable $w_{se,er,r}$ looks as follows:

$$\sum_{t \in \mathcal{T}} x_{se,t,er,r} = w_{se,er,r} \quad \forall se \in \mathcal{SE}, er \in se, r \in er \tag{4.3.17}$$

A sub-event cannot be assigned a start time if there is not enough continuous times after the start time to fulfill the duration, ensured by the constraint:

$$y_{se,t} = 0 \quad \forall se \in \mathcal{SE}, t \in \mathcal{T} \setminus t_D, \rho(t) + D_{se} - 1 > |\mathcal{T}| \tag{4.3.18}$$

Active Sub-events

As we create all possible sub-events for a given event, only a subset of these should be active in the final solution. The binary variable $u_{se} \in \{0, 1\}$ takes value 1 if sub-event $se \in \mathcal{SE}$ is active and 0 otherwise. Recall that a sub-event is active if its assigned a starting time, or if is assigned at least one non-preassigned resource. Let the parameter $PA_{er} \in \{0, 1\}$ take value 1 if event resource er has a preassigned resource, and 0 otherwise. The following constraints are imposed.

$$\sum_{r \in er \setminus r_D} w_{se,er,r} \leq u_{se} \quad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{4.3.19}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} \leq u_{se} \quad \forall se \in \mathcal{SE} \tag{4.3.20}$$

$$\sum_{t \in \mathcal{T} \setminus t_D} y_{se,t} + \sum_{r \in er \setminus r_D} w_{se,er,r} \geq u_{se} \quad \forall se \in \mathcal{SE}, er \in se, PA_{er} = 0 \tag{4.3.21}$$

Constraint (4.3.21) is necessary to ensure events are not set as active, even though they do not meet the required criteria.

The duration of active sub-events for a given event must be exactly the same as the total duration of the event (by definition of a valid XHSTT solution),

$$\sum_{se \in e} D_{se} \cdot u_{se} = D_e \quad \forall e \in \mathcal{E} \tag{4.3.22}$$

A number of constraints require that the value of a deviation $V \in \mathbb{N}$ should be within an upper-limit $\overline{B}_c \in \mathbb{N}$ and a lower-limit $\underline{B}_c \in \mathbb{N}$. This means that the penalty is defined as the amount

which the value of a deviation exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$. To simplify notation for these cases, we introduce the function $\mathcal{U}_{\underline{B}_c,\overline{B}_c}V$, which is defined as follows:

$$s \geq \mathcal{U}_{\underline{B}_c,\overline{B}_c}V \Rightarrow \begin{cases} s \geq V - \overline{B}_c \\ s \geq \underline{B}_c - V \end{cases} \tag{4.3.23}$$

Thereby the slack-variable s is forced to take the actual value of the imposed penalty.

A resource is *busy* at some time if it attends at least one solution event at that time, and busy at some time group if it is busy at one or more times within times of that time group. Let variable $q_{r,t} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time $t \in \mathcal{T}$, and 0 otherwise. Similarly, let the binary variable $p_{r,tg} \in \{0,1\}$ take value 1 if resource $r \in \mathcal{R}$ is busy in time group $tg \in \mathcal{TG}$, and 0 otherwise. The values of the two variables are determined by the following constraints.

$$|\mathcal{SE}| \cdot q_{r,t} \geq v_{t,r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{4.3.24}$$

$$q_{r,t} \leq v_{t,r} \quad \forall r \in \mathcal{R}, t \in \mathcal{T} \setminus t_D \tag{4.3.25}$$

$$p_{r,tg} \geq q_{r,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{4.3.26}$$

$$p_{r,tg} \leq \sum_{t \in tg} q_{r,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{4.3.27}$$

Constraints (4.3.24) and (4.3.26) establishes lower bounds for the variables $q_{r,t}$ and $p_{r,tg}$, i.e. ensures that these must take value 1 in case the resource is actually busy in the respective time/time group. Constraints (4.3.25) and (4.3.27) are necessary to ensure that in case the resource is in fact *not* busy in the respective time/time group, variables $q_{r,t}$ and $p_{r,tg}$ must take value 0.

In the following the constraint types of the XHSTT documentation are formulated one by one. Each constraint type is described in brief terms, and we refer to Kingston (2013c) for more details. The formulation of these constraints in terms of a Mixed-Integer Linear Programming model has not been published before. We let the 'pseudo-set' $\bar{\mathcal{C}} \subseteq \mathcal{C}$ denote constraints of a certain type depending on the context, for instance the set of all *assign resource* constraints. Furthermore we in the following make use of the general slack variable $s_{c,p,d}$, and will for each type of constraint implicitly define a corresponding slack variable with the appropriate indices for point-of-applications and deviations.

Assign Resources

Applies to: Events
Point-of-application: Event-resource
An *Assign Resource* constraint penalizes event resources that are not assigned resources. Specifically, the deviation at one point of application (an event resource with the appropriate role) is the sum of the duration of the sub-events of the respective event which are not assigned a resource. The cost of this constraint is given by:

$$D_e - \sum_{\substack{se \in e \\ r \in er \setminus r_D}} D_{se} \cdot w_{se,er,r} = s^{\text{assignres}}_{c,er} \quad \forall c \in \bar{\mathcal{C}}, e \in c, er \in e, \text{role}_{er} = \text{role}_c \tag{4.3.28}$$

Assign Time

Applies to: Events
Point-of-application: Events
The assign time constraint penalizes sub-events which are not assigned times. The deviation at one point of application is the total duration of those sub-events derived from a specific event that are not assigned a time.

$$D_e - \sum_{\substack{t \in \mathcal{T} \setminus t_D \\ se \in e}} D_{se} \cdot y_{se,t} = s^{\text{assigntime}}_{c,e} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{4.3.29}$$

Split Events

Applies to: Events
Point-of-application: Events
A *Split Event* constraint places limits on the number of sub-events that may be derived from a given event, and on their duration. Let the parameters $\underline{B}_c^{\text{amount}} \in \mathbb{N}$ and $\overline{B}_c^{\text{amount}} \in \mathbb{N}$ denote the minimum and maximum amount of sub-events which is used for a given event, respectively. And let $\underline{B}_c^{\text{dur}} \in \mathbb{N}$ and $\overline{B}_c^{\text{dur}} \in \mathbb{N}$ be the minimum and maximum duration a sub-event can have for a given event, respectively.

The cost of this constraint is given by the number of sub-events whose duration is less than $\underline{B}_c^{\text{dur}}$ or greater than $\overline{B}_c^{\text{dur}}$, and the amount by which the number of sub-events fall short of $\underline{B}_c^{\text{amount}}$ or exceed $\overline{B}_c^{\text{amount}}$. The following constraints are imposed:

$$\mathcal{U}_{\underline{B}_c^{\text{amount}},\overline{B}_c^{\text{amount}}} \sum_{se \in e} u_{se} \leq s_{c,e}^{\text{spliteventamount}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{4.3.30}$$

$$\sum_{\substack{se \in e \\ \underline{B}_c^{\text{dur}} > D_{se} \vee \overline{B}_c^{\text{dur}} < D_{se}}} u_{se} = s_{c,e}^{\text{spliteventdur}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{4.3.31}$$

The full deviation for constraint $c \in \bar{\mathcal{C}}$ and event $e \in c$ is given by $s_{c,e}^{\text{spliteventdur}} + s_{c,e}^{\text{spliteventamount}}$.

Distribute Split Event

Applies to: Events
Point-of-application: Events
The *Distribute Split Event* constraints set limits on the number of sub-events which may be derived from an event. Let $D_c \in \mathbb{N}$ be the duration of the sub-events for which this constraint applies, and let $\underline{B}_c$ and $\overline{B}_c$ be the minimum and maximum number of sub-events of duration D_c which may be derived from a given event.

$$\mathcal{U}_{\underline{B}_c,\overline{B}_c} \sum_{\substack{se \in e \\ D_{se} = D_c}} u_{se} \leq s_{c,e,er}^{\text{distsplitevent}} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{4.3.32}$$

Prefer Resources

Applies to: Events
Point-of-application: Event-resources
This constraint defines that an event resource has different preferences for certain resources. The deviation is calculated by taking all the solution resources derived from the event resource that are assigned a resource that is not one of the preferred resources, and summing the duration of the sub-events that those resources lie in. Let $r \in c$ denote a preferred resources.

$$\sum_{\substack{se \in e \\ r \notin c, r \neq r_D}} D_{se} \cdot w_{se,er,r} = s_{c,er}^{\text{preferres}} \quad \forall c \in \bar{\mathcal{C}}, e \in c, er \in e, PA_{er} = 0, \text{role}_{er} = \text{role}_c \tag{4.3.33}$$

Prefer Times constraints

Applies to: Events
Point-of-application: Events
Like the Prefer Resources constraint, events might also have preferences for certain times. The deviation is calculated for each event by summing the duration of all sub-events which is assigned a time which is not one of the preferred time. The constraint has an optional duration-property,

denoted $D_c \in \mathbb{N}_0$. If this property is given, only sub-events of duration D_c are considered. Let $t \in c$ denote a preferred time.

$$\sum_{\substack{se \in e \\ t \notin c, t \neq t_D \\ D_c = D_{se}}} D_{se} \cdot y_{se,t} = s^{\text{prefertime}}_{c,e} \quad \forall c \in \bar{\mathcal{C}}, e \in c \tag{4.3.34}$$

Avoid Split Assignments

Applies to: Evengroups
Point-of-application: Eventgroups
Each solution resource can only have one resource assigned. However, when an event is split into sub-events, each of its event resources is split into several solution resources, and a different resource may be assigned to each of these solution resources. This constraint penalizes the assignment of different resource to these solution resources. The constraint examines all the solution resources derived from those event resources, and calculates the number of distinct resources assigned to them, ignoring unassigned solution resources. The deviation is the amount by which this number exceeds 1. Let variable $k_{c,eg,r} \in \{0,1\}$ take value 1 if event e is assigned to resource r with respect to avoid split assignment constraint c, and 0 otherwise.

$$\sum_{\substack{er \in e, PA_{er}=0 \\ \text{role}_c = \text{role}_{er}}} w_{se,er,r} \leq k_{c,eg,r} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, e \in eg, se \in e \tag{4.3.35}$$

$$\sum_{r \in \mathcal{R}} k_{c,eg,r} - 1 \leq s^{\text{avoidsplitass}}_{c,eg} \quad \forall c \in \bar{\mathcal{C}}, eg \in c \tag{4.3.36}$$

Spread Events

Applies to: Eventgroups
Point-of-application: Eventgroups
The *Spread Event* constraint has a deviation for each time group $tg \in c \in \bar{\mathcal{C}}$. Let $\underline{B}_{c,tg}$ and $\overline{B}_{c,tg}$ be the minimum and maximum number of sub-events of a given event which can be placed in time group tg of constraint c, respectively. The deviation for each time group is given by the amount of which the number of sub-events for the given event which fall short of $\underline{B}_{c,tg}$ or exceeds $\underline{B}_{c,tg}$.

$$\mathcal{U}_{\underline{B}_{c,tg}, \overline{B}_{c,tg}} \sum_{\substack{se \in e \in eg \\ t \in tg}} y_{se,t} \leq s^{\text{spreadevent}}_{c,eg,tg} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, tg \in c \tag{4.3.37}$$

Link Events

Applies to: Eventgroups
Point-of-application: Eventgroups
A *Link Event* constraint specifies that some events should be assigned the same times. For each event of a given event group we build the set of times that the sub-events derived from that event are running (not just starting times). The deviation is then the number of times that appear in at least one of these sets but not in all of them. Let variable $o_{e,t} \in \{0,1\}$ take value 1 if at least one sub-event of event $e \in c \in \bar{\mathcal{C}}$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Let variable $l_{eg,t} \in \{0,1\}$ take value 1 if at least one event of event group $eg \in c$ is assigned to time $t \in \mathcal{T}$, and 0 otherwise. Constraints (4.3.38) and (4.3.40) ensure that these variables take correct values. The slack of Link Events constraints is defined in (4.3.41). Constraint (4.3.39) is necessary to restrict $o_{e,t}$ to take value 1 in cases where the event is in fact not assigned to the particular time, which would avoid the penalty given by constraint (4.3.41), if any.

$$\sum_{t' \in T^{\text{start}}_{se,t}} y_{se,t'} \leq o_{e,t} \quad \forall e \in \mathcal{E}, se \in e, t \in \mathcal{T} \setminus t_D \tag{4.3.38}$$

$$\sum_{\substack{se \in e \\ t' \in T^{\text{start}}_{se,t}}} y_{se,t'} \geq o_{e,t} \quad \forall e \in \mathcal{E}, t \in \mathcal{T} \setminus t_D \tag{4.3.39}$$

$$l_{eg,t} \geq o_{e,t} \quad \forall eg \in \mathcal{EG}, e \in eg, t \in \mathcal{T} \setminus t_D \tag{4.3.40}$$

$$l_{eg,t} - o_{e,t} \leq s^{\text{linkevent}}_{c,eg,t} \quad \forall c \in \bar{\mathcal{C}}, eg \in c, e \in eg, t \in \mathcal{T} \setminus t_D \tag{4.3.41}$$

Order Events

Applies to: Pair of events
Point-of-application: Pair of events
An *Order Event* constraint specifies that the times two events are assigned should be in order, such that the first event ends before the second event starts. Let the parameters $\underline{B}_c \in \mathbb{N}$ and $\overline{B}_c \in \mathbb{N}$ be the minimum and maximum number of times that may separate the two events, respectively. Let $(e, e') \in c$ denote an *EventPair* which this constraint applies to. Let the variable $h^{\text{last}}_e \in \mathbb{N}$ be the ordinal number of the latest time assigned to any sub-event of event e. Let the variable $h^{\text{first}}_e \in \mathbb{N}$ be the ordinal number of the first assigned to any sub-event of event e'. The deviation is then given by the amount by which the difference between these two numbers exceeds $\overline{B}_c$ or falls short of $\underline{B}_c$.

$$\rho(t) \cdot y_{se,t} + D_{se} \leq h^{\text{last}}_e \quad \forall c \in \bar{\mathcal{C}}, e \in c, se \in e, t \in \mathcal{T} \setminus t_D \tag{4.3.42}$$

$$|\mathcal{T}| - (|\mathcal{T}| - \rho(t)) \cdot y_{se,t} \leq h^{\text{first}}_e \quad \forall c \in \bar{\mathcal{C}}, e \in c, se \in e, t \in \mathcal{T} \setminus t_D \tag{4.3.43}$$

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c}(h^{\text{last}}_{e'} - h^{\text{first}}_e) \leq s^{\text{orderevents}}_{c,(e,e')} \quad \forall c \in \bar{\mathcal{C}}, (e, e') \in c \tag{4.3.44}$$

Avoid Clashes

Applies to: Resources
Point-of-application: Resources
These constraints specify that certain resources should have no clashes, i.e. they should not be assigned two or more events simultaneously. The constraint produces a set of deviations at each point of application (each resource). For each time a resource is assigned two or more solution resources, there is one deviation with a value equal to the number of solution resources minus one.

$$v_{t,r} - 1 \leq s^{\text{avoidclashes}}_{c,r,t} \quad \forall c \in \bar{\mathcal{C}}, r \in c, t \in \mathcal{T} \setminus t_D \tag{4.3.45}$$

Avoid Unavailable Times

Applies to: Resources
Point-of-application: Resource
An *Avoid Unavailable Times* constraint specifies that certain resources are unavailable for all events at certain times. The deviation is the number of unavailable times during which the resource attends at least one solution event. $t \in c$ denotes that t is an unavailable time for constraint $c \in \bar{\mathcal{C}}$.

$$\sum_{t \in c} q_{r,t} = s^{\text{unavailabletimes}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{4.3.46}$$

Limit Idle Times

Applies to: Resources
Point-of-application: Resources
A resource is idle at some time $t \in tg$ wrt. time group tg if it is not attending any sub-events at that time, but it is busy at some earlier time and at some later time in time group tg. The *Limit Idle Times* places limits on the number of idle times a resources may have. Let the variables $h^{\text{first}}_{r,tg} \in \mathbb{N}$ and $h^{\text{last}}_{r,tg} \in \mathbb{N}$ indicate the ordinal number of the first and the last time, respectively, where resource $r \in \mathcal{R}$ is busy in time group tg. Let $|tg|$ denote the amount of times in time group tg. Let the variable $h_{r,tg} \in \mathbb{N}$ denote the number of idle times of resource $r \in \mathcal{R}$ in time group $tg \in \mathcal{TG}$.

$$|tg| - (|tg| - \rho(t)) \cdot q_{r,t} \geq h^{\text{first}}_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{4.3.47}$$

$$\rho(t) \cdot q_{r,t} \leq h^{\text{last}}_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t \in tg \tag{4.3.48}$$

$$h^{\text{last}}_{r,tg} - h^{\text{first}}_{r,tg} + 1 - \sum_{t \in tg} q_{r,t} = h_{r,tg} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG} \tag{4.3.49}$$

For each resource of the constraint the deviation is calculated as follows. Calculate the total amount of idle times for all times $tg \in c$, and find the amount which this summed value falls short of minimum $\underline{B}_c \in \mathbb{N}$ or exceeds maximum $\overline{B}_c \in \mathbb{N}$. The deviation is then given by the sum of these amounts.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} h_{r,tg} \leq s^{\text{idletimes}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{4.3.50}$$

Cluster Busy Times

Applies to: Resources
Point-of-application: Resources
A *Cluster Busy Times* constraint limits the number of time groups during which a resource may be busy. The deviation is given by the amount of by which the number of given time groups during which the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum, $\overline{B}_c \in \mathbb{N}$. Let $tg \in c$ denote a time group which this constraint applies to.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{tg \in c} p_{r,tg} \leq s^{\text{clusterbusy}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{4.3.51}$$

Limit Busy Times

Applies to: Resources
Point-of-application: Resources
The *Limit Busy Times* constraints places limits on the number of times a resource may be busy within some time groups. These constraints produces a set of deviation at each point-of-application, one for each given time group. The deviations are given by the amount by which the number of times of the given time group that the resource is busy falls short of minimum, $\underline{B}_c \in \mathbb{N}$, or exceeds maximum $\overline{B}_c \in \mathbb{N}$.

$$-|tg| \cdot (1 - p_{r,tg}) + \mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{t \in tg} q_{r,t} \leq s^{\text{limitbusy}}_{c,r,tg} \quad \forall c \in \bar{\mathcal{C}}, r \in c, tg \in c \tag{4.3.52}$$

Limit Workload

Applies to: Resources
Point-of-application: Resources
A workload of a solution resource is given by $W_{e,se,er} = \frac{D_{se} \cdot L_{er}}{D_e}$, where $L_{er} \in \mathbb{N}$ is the workload of event resource er. The value is a floating-point number. A *Limit Workload Constraint* places limits on the total workload of solutions resources that certain resources are assigned to. The deviation of this constraint is the amount by which the total workload of the solution resources assigned to that resource falls short of $\underline{B}_c \in \mathbb{N}$ or exceeds $\overline{B}_c \in \mathbb{N}$, rounded up to the nearest integer.

$$\mathcal{U}_{\underline{B}_c, \overline{B}_c} \sum_{\substack{e \in c, t \in \mathcal{T} \setminus t_D \\ se \in e, er \in e}} W_{e,se,er} \cdot x_{se,t,er,r} \leq s^{\text{limitworkload}}_{c,r} \quad \forall c \in \bar{\mathcal{C}}, r \in c \tag{4.3.53}$$

4.3.3 Mixed-Integer Programming Model

Given the definitions of all constraint types of XHSTT, and their respective slack variables, the objective of the model can be stated as eq. (4.3.54), setting aside the fact that some constraints are hard-constraints and some are soft-constraints.

$$\begin{aligned} z = & f(s^{\text{assignres}}_{c,er}) + f(s^{\text{assigntime}}_{c,e}) + f(s^{\text{spliteventamount}}_{c,e} + s^{\text{spliteventdur}}_{c,e}) + f(s^{\text{distsplitevent}}_{c,e,er}) \\ & + f(s^{\text{preferres}}_{c,er}) + f(s^{\text{prefertime}}_{c,e}) + f(s^{\text{avoidsplitass}}_{c,eg}) + f(s^{\text{spreadevent}}_{c,eg,tg}) + f(s^{\text{linkevent}}_{c,eg,t}) \\ & + f(s^{\text{orderevents}}_{c,(e,e')}) + f(s^{\text{avoidclashes}}_{c,r,t}) + f(s^{\text{unavailabletimes}}_{c,r}) + f(s^{\text{idletimes}}_{c,r}) + f(s^{\text{clusterbusy}}_{c,r}) \\ & + f(s^{\text{limitbusy}}_{c,r,tg}) + f(s^{\text{limitworkload}}_{c,r}) \end{aligned} \tag{4.3.54}$$

The full MIP would therefore consists of minimizing z, subject to eqs. (4.3.13) to (4.3.53). However, we take a different approach, as described in the next section.

4.3.4 Solution Approach

Even though it would be natural to simply input the MIP to a generic solver, a different approach is taken, which takes advantage of the XHSTT objective function. In this approach, the model is solved in two steps, denoted Step 1 and Step 2 in the following.

By the definition of the XHSTT objective, hard constraints always take priority over soft constraints. Therefore the following approach is taken for solving the model: In Step 1, a MIP is build which only contains the hard constraints. This MIP is given as input to the MIP solver, which is ran until the given time limit is reached, or until the model is solved to optimality. The found objective value is the hard cost of the solution. In case the time limit is reached, all variables are fixed to their final value (i.e. the value they take in the best found solution), and all the soft constraints are added to identify the true cost of the found solution. In case the MIP is solved to optimality, Step 2 is performed: All soft constraints are added and the solution process is warm-started from its previous state, with the time limit set to what remains of the original time limit. Furthermore a constraint is added which ensures that the optimal value of the hard cost is kept. Let z^{hard} denote the sum of all slack variables belonging to the hard constraints. The following constraint is added:

$$z^{\text{hard}} = \text{hard cost} \tag{4.3.55}$$

Now this extended MIP model is solved. The cost of the obtained solution, minus the hard cost found in Step 1, is the value of the soft cost. Notice that the nature of this solution method resembles *lexicographic multi-objective optimization.*

This approach takes advantage of the capability of MIPs to issue certificates of optimality. By this we mean that focus is put on the hard constraints until a solution is found with the optimal

hard cost, and then we switch focus and consider the entire problem instance. If a heuristic solution method was used the inevitable question would be: When has sufficient effort been put into minimizing the hard cost?

4.4 Computational Results

This section presents computational results of the developed exact method, and has two primary intentions:

- How does the MIP compete with the heuristics of the ITC2011 round 2? Thereby the potential of this MIP approach can be evaluated on fair terms with well-performing heuristics. This is the subject of Section 4.4.1.
- Are we able to improve the best-known solutions for some instances, or even solve them to optimality? See Section 4.4.2.

All tests were run on a machine with an Intel i7 CPU clocked at 2.80 GHz and 12GB of RAM, running Windows 8 64 bit. In all cases the commercial state-of-art MIP solver Gurobi 5.5.0 was used. Two distinct sets of XHSTT instances have been used, both obtained from the XHSTT website (Post, 2013b). All obtained solutions have been verified as being valid using the evaluator *HSEval* (Kingston, 2013b).

As described in Section 4.3.1, an XHSTT objective consists of both a hard cost, and a soft cost, usually denoted (hard cost, soft cost). In case a solution has a hard cost of value 0, the objective is simply written as the soft cost, as is usually done in context of the XHSTT format.

As discussed in Section 4.3.1, the size of the sets $\mathcal{I}$ and $\mathcal{J}$ must be selected sufficiently high. Notice further that if the size of these sets is selected high, it can have a big impact on the amount of variables in the model. It would be possible to select these sizes based on the properties of the constraints having CostFunction `SumSquare` or `SquareSum`, however this is a quite complex operation as it must be derived based on each constraint-type. Instead we have selected $|\mathcal{I}| = 10$ $|\mathcal{J}| = 10$, such that the maximum possible penalty is $9^2 = 81$. This means that we cannot claim optimality for solutions with objective > 81. An easy fix for this issue is to simply perform a re-run of Gurobi if a solution is claimed as optimal, with the size of the sets set to a higher value. The same is applicable for lower bounds of value > 81. We consider this is an implementation detail, and it will be seen that in practice it has no impact of the obtained results.

4.4.1 International Timetabling Competition 2011

This section compares the exact method with the results obtained by the finalists in Round 2 of ITC2011. In this round the solver for each participating team was tested on 18 previously unknown instances from the archive `XHSTT-ITC2011-hidden`. The time limit for all instances was nominated to 1000 seconds, but the organizers provided a tool to benchmark machines to find the machine-dependent equivalent of this time limit. On our machine this amended to 772 seconds. The possibility to benchmark machines facilitates a fair comparison with the competitors of ITC2011, except for the fact that the rules of ITC2011 did not allow the use of commercial software, which conflicts with our use of Gurobi. The aim of this section is therefore to demonstrate the potential of MIP in the context of timetabling (which is often overlooked), and not to claim how our approach would have positioned itself in ITC2011.

In terms of solver parameters, default settings are used, except for the *pseudo-parameter* `MIPFocus` which is set to value 1, emphasizing that we are mainly interested in finding incumbent solutions. Gurobi was only allowed to use a single CPU thread, as specified in the rules of ITC2011.

The participants of ITC2011 round 2 ran their algorithm 10 times on each instance, to eliminate the stochastic impact on the results. Since we are interested in the average performance of each participant for comparison, the following processing of the results was performed: For each instance and each participant, calculate both the average hard cost and the average soft cost, and round

Table 4.1: Performance of the MIP using same running time as specified in ITC2011. For each instance is listed the average solution found from each of the competitors of ITC2011, and the solution obtained by the MIP formulations. The best solutions are marked in **bold**. Objectives marked with * are optimal solutions.

Instance	GOAL	HySST	Lectio	HFT	Exact method
BrazilInstance2	(1, 62)	(1, 77)	**38**	(6, 190)	46
BrazilInstance3	124	118	152	(30, 283)	**39**
BrazilInstance4	(17, 98)	(4, 231)	**(2, 199)**	(67, 237)	(5, 286)
BrazilInstance6	(4, 227)	(3, 269)	**230**	(23, 390)	682
ElementarySchool	4	(1, 4)	**3**	(30, 73)	**3**
SecondarySchool2	**1**	23	34	(31, 1628)	(1604, 3878)
Aigio	**13**	(2, 470)	1062	(50, 3165)	(1074, 3573)
Italy_Instance4	**454**	6926	651	(263, 6379)	17842
KosovaInstance1	**(59, 9864)**	(1103, 14890)	(275, 7141)	(989, 39670)	(3626, 2620)
Kottenpark2003	**90928**	(1, 56462)	(50, 69773)	(209, 84115)	(8491, 6920)
Kottenpark2005A	**(31, 32108)**	(32, 30445)	(350, 91566)	(403, 46373)	(2567, 53)
Kottenpark2008	**(13, 33111)**	(141, 89350)	(209, 98663)	-	(14727, 5492)
Kottenpark2009	**(28, 12032)**	(38, 93269)	(128, 93634)	(345, 99999)	(17512, 140)
Woodlands2009	(2, 14)	(2, 70)	**(1, 107)**	(62, 338)	(1801, 705)
Spanish school	**894**	1668	2720	(65, 13653)	(1454, 11020)
WesternGreece3	**6**	11	(30, 2)	(15, 190)	25
WesternGreece4	**7**	21	(36, 95)	(237, 281)	81
WesternGreece5	**0**	4	(4, 19)	(11, 158)	15
Avg. Ranking	1.72	2.67	2.50	4.44	3.61

both to nearest integer. These numbers then denote how this participant performed on this instance.

Table 4.1 shows the obtained results. The value of "Avg. Ranking" was calculated as follows. Each solution method was ranked 1 to 5 on each instance, 1 being the best, and the average of these ranks was taken. According to this measure, the exact method of this paper is competitive with the methods used at ITC2011. Notice in particular that the exact method performs well on the smaller instances, and is generally not as competitive on the larger instances. On three instances the exact method gave the best results.

4.4.2 Aiming at Optimality

In attempt to produce new (optimal) solutions, the XHSTT archive `ALL_INSTANCES` was used, which contains 38 non-artificial instances. According to the website, this archive "contains all latest versions of the contributed instances". For 10 of the instances, a solution with cost 0 is already known, which constitutes an optimal solution by the definition of XHSTT. Hence these instances are skipped in this test. Notice that `ALL_INSTANCES` contains instances which originally came from `XHSTT-ITC2011-hidden`, but due to bug-fixes in some of the instances, we consider them as two separate sets of instances (by bug-fixes we mean altering of certain constraints, such that objective values are incomparable). We refer to (Post, 2013b) for instance-statistics.

This test was performed with the following setup: Gurobi is allowed to use all CPU cores (which is 8 in our case), and the time-limit is set to 24 hours for each instance. As initial solution for each instance, the current best known solution is provided. Default parameter settings of Gurobi were used. Table 4.2 shows the obtained results. A gap between an incumbent solution x and a lower bound LB is calculated by $\frac{|x-\text{LB}|}{x}$.

For each instance a solution with XHSTT objective (H, S) is found, as well as a lower bound $(\underline{H}, \underline{S})$. By the definition of our solution method, we only have a lower bound on the soft cost $\underline{S}$ iff an optimal solution for the hard cost is known, i.e. $H = \underline{H}$. If a lower bound or an objective value is not found we write "-". Notice that even though we give the current best known solution as starting solution, Gurobi might still not find a solution for Step 1, usually in case the instance in question is of huge size. In Table 4.2, both the gap for the hard cost and the soft cost is shown (in case the required costs and lower bounds are available).

Table 4.2 shows that our method obtains better solutions for 8 instances. 4 instances was solved to optimality, proving optimality of 3 previously known solutions and finding 1 new optimal

Table 4.2: Performance of the MIP on `ALL_INSTANCES`. For each instance is listed the best previously known solution *"Best"*, and for the solution found by our approach is listed the time used to solve Step 1 *"$Time_1$"*, the time used to solve Step 2 *"$Time_2$"*. *"Time"* indicates the total solving time. All times have seconds as unit. Furthermore the objective *"Obj"* and the lower bound *"LB"* is listed. The percentage gap between the objective and the lower bound is divided into the gap for the hard constraints *"Gap_1"* and the gap for the soft constraints, *"Gap_2"*. Objectives in **bold** denote new best solution while optimal solutions are marked with *.

			MIP solution method						
	Instance	Best	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
AU	BGHS98	(3, 494)	>86400	-	>86400	(3, 494)	(-,-)	-	-
AU	SAHS96	(8, 52)	>86400	-	>86400	(8, 52)	(-,-)	-	-
AU	TES99	(1, 140)	>86400	-	>86400	(1, 140)	(0,-)	100.0	-
BR	Instance1	42	0	>86400	>86400	**40**	28	0.0	30.0
BR	Instance2	5	1	>86399	>86400	5	1	0.0	80.0
BR	Instance3	47	1	>86399	>86400	**26**	19	0.0	26.9
BR	Instance4	78	1	>86399	>86400	**61**	42	0.0	31.2
BR	Instance5	43	1	>86399	>86400	**30**	10	0.0	66.7
BR	Instance6	60	1	>86399	>86400	60	14	0.0	76.7
BR	Instance7	122	1	>86399	>86400	122	22	0.0	82.0
DK	Falkoner2012[1]	(2, 23705)	>86400	-	>86400	(2, 23705)	(0,-)	100.0	-
DK	Hasseris2012[2]	(293, 32111)	>86400	-	>86400	(293, 32111)	(-,-)	-	-
DK	Vejen2009[3]	(20, 18966)	>86400	-	>86400	(20, 18966)	(2,-)	90.0	-
UK	StPoul	136	52	>86348	>86400	136	0	0.0	100.0
FI	ElementarySchool	3	2	785	787	*3	3	0.0	0.0
FI	HighSchool	1	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	1	>86399	>86400	88	77	0.0	12.5
GR	UniInstance3[4]	5	0	3	3	*5	5	0.0	0.0
GR	UniInstance4[5]	8	1	>86399	>86400	8	0	0.0	100.0
IT	Instance1	12	1	4561	4562	*12	12	0.0	0.0
IT	Instance4	78	12	>86389	>86400	**62**	27	0.0	56.5
XK[6]	Instance1	3	31	>86369	>86400	3	0	0.0	100.0
NL	GEPRO	(1, 566)	>86400	-	>86400	(1, 566)	(0,-)	100.0	-
NL	Kottenpark2003	1410	57	>86343	>86400	1410	(0,-)	0.0	-
NL	Kottenpark2005	1078	88	>86312	>86400	1078	9	0.0	99.2
NL	Kottenpark2009	9250	92	>86308	>86400	**9035**	160	0.0	98.2
ZA	Woodlands2009	2	22	77878	77900	***0**	0	0.0	0.0
ES	School	(3, 5966)	6525	>79875	>86400	**357**	322	0.0	9.8

[1] Shorthand for instance *FalkonergaardenGymnasium2012*
[2] Shorthand for instance *HasserisGymnasium2012*
[3] Shorthand for instance *VejenGymnasium2009*
[4] Shorthand for instance *WesternGreeceUniversityInstance3*
[5] Shorthand for instance *WesternGreeceUniversityInstance4*
[6] Kosova.

solution. Furthermore, 11 new non-trivial lower bounds and 7 new best solutions have been established for the instances which were not solved to optimality.

Alternative Formulation

The Limit Idle Times constraint is known to be difficult for solvers to handle (Dorneles et al. (2012)). In our formulation, this constraint is formulated using Big-M notation (constraints (4.3.47) and (4.3.48)), which can provide bad LP-relaxation, which in turn might slow down the solution process. Furthermore this constraint is part of most instances (29 of 38 instances in the `ALL_INSTANCES` archive), so an alternate formulation is proposed. The alternate formulation uses variable $h_{r,tg,t} \in \{0,1\}$ which takes value 1 if resource $r \in \mathcal{R}$ has an idle time in time $t \in tg$ in time group tg, and 0 otherwise. Constraints (4.3.47), (4.3.48) and (4.3.49) are replaced by

$$q_{r,t'} - q_{r,t} + q_{r,t''} - 1 \leq h_{r,tg,t} \quad \forall r \in \mathcal{R}, tg \in \mathcal{TG}, t, t', t'' \in tg, \rho(t') < \rho(t) < \rho(t'') \qquad (4.4.3)$$

This yields more rows in the MIP; for each time group $tg \in \mathcal{TG}$ the amount of additional constraints is $\binom{|tg|}{3}$. Furthermore, there is a great possibility that the amount of variables increases due to the extra dimension on the h variable. However, no Big-M notation is used.

Due to the possible big increase in the size of the model, this alternative formulation is only tested on the smaller instances from archive `ALL_INSTANCES`, skipping those instances in which the

Table 4.4: Performance of the alternative formulation on the smaller instances of archive `ALL_INSTANCES`. All the columns are defined in analogous way to Table 4.2, except for *"$Obj_{T4.2}$"* and *"$LB_{T4.2}$"* which denote the objective value and the lower bound found in Table 4.2.

					MIP alternative formulation						
	Instance	Best	$Obj_{T4.2}$	$LB_{T4.2}$	$Time_1$	$Time_2$	Time	Obj	LB	Gap_1	Gap_2
BR	Instance1	42	40	28	0	1918	1918	***38**	38	0.0	0.0
BR	Instance2	5	5	1	0	290	290	*5	5	0.0	0.0
BR	Instance3	47	26	19	1	>86399	>86400	**23**	21	0.0	8.7
BR	Instance4	78	61	42	1	>86399	>86400	61	49	0.0	19.7
BR	Instance5	43	30	10	1	>86399	>86400	**26**	15	0.0	42.3
BR	Instance6	60	60	14	1	>86399	>86400	**59**	18	0.0	69.5
BR	Instance7	122	122	22	1	>86399	>86400	**84**	26	0.0	69.1
FI	HighSchool	1	1	0	1	>86399	>86400	1	0	0.0	100.0
FI	SecondarySchool	88	88	77	1	>86399	>86400	**84**	77	0.0	8.3
GR	UniInstance4[1]	8	8	0	1	>86399	>86400	8	0	0.0	100.0
IT	Instance4	78	62	27	6	>86394	>86400	**57**	27	0.0	52.6
ES	School	(3, 5966)	357	322	44	>86356	>86400	357	330	0.0	7.6

[1] Shorthand for instance *WesternGreeceUniversityInstance4*

optimal solution was found in the previous test (Table 4.2). Since the goal is to achieve is good solutions as possible, we restart the procedure from the best found solution of Table 4.2 and run it for additional 24 hours. This test-setup means that we cannot compare the performance of the two formulations. Table 4.4 shows the obtained results. The table shows that this formulation is capable of finding 2 new optimal solutions. For the instances not solved to optimality, 6 lower bounds were improved, and new best solutions were found for 6 instances.

4.5 Conclusion

This paper has shown the first exact method for (High) School Timetabling instances in the XHSTT format. A solution method which takes advantage of the structure of the objective function of XHSTT has been proposed. For the most recent version of the archive `ALL_INSTANCES`, we were able to produce 2 new optimal solutions and prove optimality of 4 previously known solutions. For 11 other instances, new non-trivial lower bounds were shown. For the instances not solved to optimality, we were able to improve the best known solution in 9 cases.

Establishing optimal solutions and lower bounds is indeed a step forward for research within high school timetabling, and for the XHSTT format in particular. This gives researchers a possibility to compare their obtained solutions with an (optimal) lower bound, which is valuable for evaluating the quality of solutions.

As subjects for future research the following are mentioned. The MIP could be used in context of *Two-Stage Decomposition* (TSD), by first assigning times to events, and secondly assigning resources to event resources. Thereby the resource-assignments are done subject to the times assigned to events. Such an approach was used with great success in the paper of Lach and Lübbecke (2012) for the *Curriculum-based University Timetabling Problem* (the optimization problem used in the International Timetabling Competition 2007), and by Sørensen and Dahms (2014) for the real-world case of High School Timetabling in Denmark. In both of these papers, the TSD is theoretically capable of producing near-optimal results, even though the problem is split into two separate MIPs. However, the XHSTT case is possibly less suited for this type of decomposition as instances might contain a majority of constraints related to resource assignments. Since the assignments to times for events are performed in the first stage of the decomposition, and because these assignments cannot be altered when the resource-assignments are performed, a TSD approach would possibly be heuristic in nature. Obviously, if an XHSTT instance have all resources preassigned to event resources, a TSD would be unnecessary.

Our MIP formulation is exponential in size by the amount of sub-events in the instance, as all possible combinations of sub-events are generated. A better formulation would be less dependent on this amount. One could for instance solve the model iteratively, and 'inject' new sub-events

in the model on-the-fly. Another possibility would be to consider a formulation which simulate sub-events by an integer variable which define the lengths of each respective active sub-event. Such improved formulations are subject for future research.

Bibliography

P. Avella, B. D'Auria, S. Salerno, and I. Vasilâev. A computational study of local search algorithms for italian high-school timetabling. *Journal of Heuristics*, 13:543–556, 2007. ISSN 1381-1231.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

R. E. Bixby. *Optimization Stories*, volume Extra of *21st International Symposium on Mathematical Programming Berlin*, chapter A Brief History of Linear and Mixed-Integer Programming Computation, pages 107–121. Journal der Deutschen Mathematiker-Vereinigung, August 19–24 2012.

M. Ter Braak. A hyperheuristic for generating timetables in the xhstt format. Master's thesis, University of Twente, June 2012.

Á. P. Dorneles, O. C. de Araújo, S. Maria-Brazil, and L. S. Buriol. The impact of compactness requirements on the resolution of high school timetabling problem. In *Congreso Latino-Iberoamericano de Investigación Operativa*, September 2012.

G. Fonseca, H. Santos, T. Toffolo, S. Brito, and M. Souza. A sa-ils approach for the high school timetabling problem. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

A. Kheiri, E. Ozcan, and A. J. Parkes. Hysst: Hyper-heuristic search strategies and timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 497–499, 2012.

J. H. Kingston. Educational timetabling. In A. S. Uyar, E. Ozcan, and N. Urquhart, editors, *Automated Scheduling and Planning*, volume 505 of *Studies in Computational Intelligence*, pages 91–108. Springer Berlin Heidelberg, 2013a. ISBN 978-3-642-39303-7.

J. H. Kingston. The hseval high school timetable evaluator. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi` [Retrieved 28/11-2013], Aug. 2013b.

J. H. Kingston. High school timetable file format specification: Constraints. `http://sydney.edu.au/engineering/it/~jeff/hseval.cgi?op=spec&part=constraints` [Retrieved 28/11-2013], Aug. 2013c.

G. Lach and M. Lübbecke. Curriculum based course timetabling: new solutions to udine benchmark instances. *Annals of Operations Research*, 194:255–272, 2012. ISSN 0254-5330.

M. Pimmer and G. R. Raidl. A timeslot-filling heuristic approach to construct high-school timetables. In L. Di Gaspero, A. Schaerf, and T. Stützle, editors, *Advances in Metaheuristics*, volume 53 of *Operations Research/Computer Science Interfaces Series*, pages 143–157. Springer New York, 2013. ISBN 978-1-4614-6321-4.

G. Post. International timetabling competition 2011 results. `http://www.utwente.nl/ctit/hstt/itc2011/results/` [Retrieved 28/11-2013], Aug. 2013a.

G. Post. Benchmarking project for (high) school timetabling. `http://www.utwente.nl/ctit/hstt/` [Retrieved 28/11-2013], Aug. 2013b.

G. Post, S. Ahmadi, S. Daskalaki, J. Kingston, J. Kyngas, C. Nurmi, and D. Ranson. An xml format for benchmarks in high school timetabling. *Annals of Operations Research*, 194:385–397, 2012a. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012b.

J. Romrös and J. Homberger. An evolutionary algorithm for high school timetabling. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 485–488. SINTEF, 2012.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

M. Sørensen and F. H. W. Dahms. A two-stage decomposition of high school timetabling applied to cases in denmark. *Computers & Operations Research*, 43:36–49, March 2014.

M. Sørensen and T. R. Stidsen. Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. *Annals of Operations Research*, PATAT 2012 SI:Submitted Jan 21. 2013, 2013.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

C. Valouxis, C. Gogos, P. Alefragis, and E. Housos. Decomposing the high school timetable problem. In *Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

Part III

Student Sectioning Problems at Danish High Schools

Chapter 5

Elective Course Planning

Simon Kristiansen*† Matias Sørensen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, msso@dtu.dk, thst@dtu.dk

† MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

Abstract Efficient planning increasingly becomes an indispensable tool for management of both companies and public organizations. This is also the case for high school management in Denmark, because the growing individual freedom of the students to choose courses makes planning much more complex. Due to reforms, elective courses are today an important part of the curriculum, and elective courses are a good way to make high school education more attractive for the students. In this article, the problem of planning the elective courses is modeled using integer programming and three different solution approaches are suggested, including a Branch-and-Price framework using partial Dantzig-Wolfe decomposition. Explicit Constraint Branching is used to enhance the solution process, both on the original IP model and in the Branch-and-Price algorithm. To the best of our knowledge, no exact algorithm for the Elective Course Planning Problem has been described in the literature before. The proposed algorithms are tested on data sets from 98 of the 150 high schools in Denmark. The tests show that for the majority of the problems, the optimal solution can be obtained within the one hour time bound. Furthermore the suggested algorithms achieve better results than the currently applied meta-heuristic.

5.1 Introduction

Management of high schools in Denmark has become increasingly complex during the last decade, due to a number of economical and educational reforms. Furthermore the high schools, 10th to 12th grade, became self-governing institutions as of 1/1 2007. Hence the administration of a high school in Denmark has lately been given much more freedom to manage the high school. On the other hand, the high school management now has a much higher economical responsibility and the high schools can even become insolvent, which has indeed happened in a few cases. The by far most important income for the high schools is the grant received from the government. Due to the recent reforms, high schools receive this grant based on the number of students graduating. This has forced the high school management to focus on attracting students, while reducing teaching

[1] Published in *European Journal of Operational Research, volume 215, 2011*

and administration costs. In this paper we present the *Elective Course Planning Problem* (ECPP), which is of crucial importance for maintaining student contentment.

Due to the special structure of the Danish high schools, literature concerning the ECPP is very limited. However, problems such as *Course Timetabling* and *Student Sectioning* have been looked in to, see e.g. Tripathy (1984); Laporte and Desroches (1986); Erben and Keppler (1996); Rudova and Murray (2003); Müller and Murray (2010). The problem in Tripathy (1984) is actually closely related to the ECPP of this article, but does however lack many of the relevant constraints.

The *International Timetabling Competition 2007* had two tracks focusing on course timetabling, *Post Enrolment based Course Timetabling* and *Curriculum based Course Timetabling*. The first track considers the problem of construction a timetable according to the choice of lectures of the students (Lewis et al. (2007); Müller (2009)), while the second considers weekly scheduling of lectures within a given number of rooms and time periods (Gaspero et al. (2007); Müller (2009)). Neither of these problems matches the ECPP, which is the problem of fulfilling as many elective course requests as possible. I.e. the ECPP is a matter of deciding which course requests should be fulfilled, while the course timetabling problems of ITC2007 are concerned with constructing a timetabling according to some predefined assignments to courses. Still, the ECPP does in fact share several constraints with the mentioned problems, e.g. students can only attend one course at a time. But as will be described in Section 5.2, the ECPP includes other important constraints which are not part of any of the tracks of ITC2007. The timetabling part of the ECPP lies in the fact that, even though the objective is to maximize the number of fulfilled course requests, the course requests should be able to fit in a number of predefined time-blocks.

Both de Werra (1985) and Costa (1994) deals with the problem of first assigning teachers to classes and then assigning classes to a schedule. However, in the Danish education system, the ECPP must be dealt with before allocating teachers to classes. Müller et al. (2007) combines University Course Timetabling and Student Sectioning in an online system based on an *Iterative Forward Search* algorithm. Neither this combination of the two problems proves very resourceful for the ECPP. There exist more practical based publications on solving problems concerning Course Timetabling and Student Sectioning, but it seems none of these are directly related to the ECPP.

Binzer and Kjeldsen (2008) is by far the most relevant source for this article, since it describes the ECPP in detail, and solves the problem using both *Tabu Search* and *GRASP*. Both these meta-heuristics rely on a complicated neighborhood, based on a number of simple moves. Binzer and Kjeldsen (2008) points to the fact that the ECPP has features usually discarded in timetabling literature. For instance, the ECPP is concerned with both optional assigning to time slots and division of classes. Most literature is only concerned with one of these aspects.

Notice that Binzer and Kjeldsen (2008) applies heuristics. In this paper, exact solution methods will be attempted. To the best of our knowledge, no exact algorithm for the ECPP has been described in the literature before.

There are two reasons why solving the ECPP to optimality is so crucial to the high schools. First of all, each offered elective class costs app. 200,000 DKK (28,500 EUR) per year to complete. These costs are fixed costs to be paid, no matter how many students are actually enrolled in the course. If the school has too many courses running with a low number of students it will hence inflict a heavy financial burden on the school. Secondly, if a student is not granted its elective course requests, the student may decide to switch to another high school, which is highly undesirable for the high school administration. Given these important trade offs it is natural that the high schools devote significant resources to find good elective course plans. This is either done manually (typically requiring 2 weeks of full time work) or using existing software solutions. Today most of the high schools (87%) use the cloud-based MaCom Lectio software system for all administrative purposes, and Lectio includes a solver for the ECPP, which is based on the meta-heuristics from Binzer and Kjeldsen (2008). It should be mentioned that the solver in Lectio is highly effective compared to the approach of solving it manually, however in this paper we will show that better algorithms can be found.

The structure of this article is as follows: The ECPP is described in detail in Section 5.2. In

Section 5.3 a MIP model is presented. In Section 5.4 the ECPP MIP model is Dantzig-Wolfe decomposed and a Branch-and-Price algorithm is designed. Furthermore, Section 5.4 presents an alternative approach, Explicit Constraint Branching. In Section 5.5 the algorithm is compared to an existing meta-heuristic for the ECPP, on real-life data from 98 Danish high schools. Finally a conclusion is attempted in Section 5.6.

5.2 Problem Description

In Denmark there exists several types of upper secondary educations. The focus in this paper is STX (Upper Secondary School Leaving Examination). STX is a broad general education and it is the most common type of secondary education in Denmark. The education takes 3 years and consists of 13 to 16 courses.

Once a student has chosen a type of upper secondary education, he needs to choose a study line which fit his interests. Different high schools offer different study lines, so the students choice of high school is somewhat correlated with the choice of study line. It is important to notice that the students can choose free of charge among all Danish high schools. Each study line consists of both a number of mandatory courses and a number of elective courses. When the student has chosen a study line, he is assigned a so-called common class. A common class always consists of students from the same study line. The idea is that courses should be taught in classes with as many students as possible from the same common class, to facilitate the cooperation and social interaction between students.

It should be noted that some courses have a duration of more than one year. This yields many pre-assignment of students to classes, which from this point on will be known as existing classes.

The elective courses give the students some influence on the contents of their education. An elective course can either be an upgrade of a mandatory course to a higher educational level, or it can be a course with a subject the students has not been taught before. Furthermore the students have the opportunity to select an elective course as their second priority. I.e. if an elective course request is not granted, it is possible to assign the student to his second priority course instead. The priority of elective course requests is however omitted in this article as the test data did not contain information regarding priority. A solution-approach for the prioritized requests would be to weight the requests based on their priority. In this article all requests are equally weighted.

5.2.1 Weekly Schedule

In a typical weekly schedule for a high school, each day consists of four modules where teaching is performed. The ECPP is concerned with assigning so-called blocks to modules in the weekly schedule. Blocks consist of a given number of classes teaching specific courses. The students cannot be taught two courses simultaneously, so they must only be part of one class in each block. Binzer and Kjeldsen (2008) refers to blocks as *finished chunks of a time plan*, which seems as a proper description. Figure 5.1 illustrates how five different blocks are assigned to modules.

	Monday	Tuesday	Wednesday	Thursday	Friday
8.15 9.45	BLOCK1			BLOCK3	
10.00 11.30					BLOCK5
	Lunch break				
12:00 13:30					
13:45 15:15		BLOCK2		BLOCK4	

Figure 5.1: Assigning blocks to modules in a weekly schedule

The elective courses are placed in special blocks in the weekly schedule, e.g. the grey blocks in Figure 5.1. The mandatory courses will hence be taught in the white blocks in the schedule of Figure 5.1. As teaching of the mandatory courses and elective courses can be said to be independent in the weekly schedule, we will in the remainder of the article ignore the problem of planning the mandatory courses.

Ideally all elective course requests should be fulfilled. However this is usually unrealistic, not only because of a number of resource limitations of the school, but also due to regulations enforced by education policies. Some common resources limitations are:

- Availability of classrooms. The number of classrooms at the high school is limited. Furthermore some course subjects may require special classrooms, e.g. physics and music education.
- Availability of teachers. It is obvious that a block cannot contain more physics classes than the number of physics teachers available. Furthermore teachers are limited to a certain number of teaching lessons per week.
- Limitations on class sizes.

To model the resource limitations, the concept of subjects is introduced. Let each course be associated with exactly one subject. Resource limitations can then be modeled by specifying a maximum number of courses of a given subject in a block. It is hence assumed that no resource is shared between subjects. This assumption is not too realistic, however it is necessary to compare our results with the existing meta-heuristic algorithm. In a better modeling approach, constraints for each resource should be part of the model. Although it should be noted that only few of the 98 obtained datasets actually specifies this maximum number of classes for any subject, so the negative impact of this assumption is limited.

5.2.2 The Problem

The objective of the ECPP can be stated as follows: Fulfill as many elective course requests as possible using a minimum number of blocks and a minimum number of classes, and assign all existing classes to a block. This suggests that the problem is multi objective, where each of the following is part of the objective:

- Minimize the number of created classes.
- Minimize the number of blocks used for elective courses.
- Maximize the number of elective course requests granted.

Due to compatibility with Binzer and Kjeldsen (2008), it is chosen to keep both the number of created classes and number of blocks constant. Hence the objective is to maximize the number of fulfilled student requests, given a choice of number of classes and blocks. A subject for future research would be to model this with multiple objectives.

Due to resource limitations and limits on the both number of blocks and classes, our mathematical formulation will not guarantee that students are granted the required courses. Consider the following case. A majority of the students in their first school year needs to choose either German B or French B. If however only a single student chooses French B, then this student will in practice never be granted his request, as no high schools will create a class with just one student. The high school has a number of options in this scenario:

- Convince the student to select another course, e.g. German B, instead.
- Convince the student to select a different study line where French B is mandatory.
- Assign the student to a class teaching French B, but where all the other students are from a different common class. Possibly even violate the upper limit on class size to fit the student in.

- The student chooses another high school, which is able to fulfill his requests.

All of these options have a downside which we will not discuss. The point is that these special scenarios cannot be taken into account by a mathematical model, as the possibilities are very diverse and wide-ranging.

5.3 Modeling of Elective Course Planning

The ECPP is now formulated as an IP model which aims at maximizing the number of granted course requests while respecting the conditions. For a particular high school there is a set of students $s \in S$, a set of courses offered $c \in C$ and a set of blocks $b \in B$. Each course belongs to one of the subjects $f \in F$. The maximum number of classes of each subject f in a block is given by $SM_f \in \mathbb{Z}^+$. The relationship between course c and subject f is defined by $SC_{c,f}$. Each student chooses a set of elective courses which he or she wishes to follow, given by $V_c^s \in \{0,1\}$ which takes value 1 if student s has chosen course c, and zero otherwise. For each class there is a lower bound and an upper bound on the number of students, $L_c \in \mathbb{Z}$ and $U_c \in \mathbb{Z}$ respectively. Naturally it applies that $L_c \leq U_c$. The number of classes which can be established is given by the number Q. Finally, there is a set of existing classes $t \in T$. Each existing class contain a set of students $E^{s,t} \in \{0,1\}$, which takes value 1 if student s is part of existing class t, and 0 otherwise. $H_c^t \in \{0,1\}$ is 1 if existing class t is teaching course c. The decision whether a student s should be assigned to course c in a block b, is defined by the binary variable $x_{c,b}^s$. The number of necessary elective course classes to form in block b of course c is given by the integer variable $y_{c,b} \in Z^+$. Finally, the variable u_b^t takes value 1 if existing class t is placed in block b, and 0 otherwise.

The entire MIP model for the ECPP is given in model (5.3.1).

IP Model for the ECPP (5.3.1)

$$\max \quad \sum_{c,b,s} x_{c,b}^s \tag{5.3.1a}$$

$$\text{s.t.} \quad \sum_{c} x_{c,b}^s + \sum_{t} E^{s,t} \cdot u_b^t \leq 1 \qquad \forall\, b, s \tag{5.3.1b}$$

$$\sum_{b} x_{c,b}^s \leq V_c^s \qquad \forall\, c, s \tag{5.3.1c}$$

$$\sum_{s} x_{c,b}^s \geq L_c \cdot y_{c,b} \qquad \forall\, c, b \tag{5.3.1d}$$

$$\sum_{s} x_{c,b}^s \leq U_c \cdot y_{c,b} \qquad \forall\, c, b \tag{5.3.1e}$$

$$\sum_{c,b} y_{c,b} \leq Q \tag{5.3.1f}$$

$$\sum_{c} SC_{c,f} \cdot y_{c,b} + \sum_{c,t} SC_{c,f} \cdot H_c^t \cdot u_b^t \leq SM_f \qquad \forall\, b, f, SM_f > 0 \tag{5.3.1g}$$

$$\sum_{b} u_b^t = 1 \qquad \forall\, t \tag{5.3.1h}$$

$$x_{c,b}^s \in \{0,1\} \tag{5.3.1i}$$

$$y_{c,b} \in \mathbb{N}_0 \tag{5.3.1j}$$

$$u_b^t \in \{0,1\} \tag{5.3.1k}$$

The objective simply sums up the number of student requests it was possible to satisfy. The constraint (5.3.1b) ensures that no student is taught simultaneously in two classes, no matter if the classes are elective classes or existing classes. Constraint (5.3.1c) ensures that a student is only granted an elective course if requested, and each request is only granted once. The constraints (5.3.1d) and (5.3.1e) sets the lower and upper bound respectively on the number of students in elective course classes. Constraint (5.3.1f) limits the maximal number of elective course classes which can be offered. Constraint (5.3.1g) ensures that the resource limit on subject f is respected.

Finally constraint (5.3.1h) ensures that all existing classes are placed in a block.

The ECPP has been proven $\mathcal{NP}$-hard with a $\mathcal{NP}$-complete decision problem in both Binzer and Kjeldsen (2008) and, in a slightly varied form, in Kristiansen and Sørensen (2010), and the problem contains more than 150.000 binary variables in several of the instances tested. Furthermore the model also contains a great deal of symmetry, since blocks are interchangeable and students who request identical courses are also interchangeable. Notice that all requests provide the same contribution to the objective when granted. See Margot (2003) and Kaibel et al. (2007) regarding solving large integer linear programs with much symmetry. We can not expect that we are able to solve the model with a standard IP solver for problem of non-trivial size.

5.4 Solution algorithms

In this section we will describe a Dantzig-Wolfe decomposition of the ECPP and an alternative approach where Explicit Constraint Branching is applied. Dantzig-Wolfe decomposition is a well-known approach applied to hard optimization problems (see e.g. Vanderbeck (2000)). For an introduction to Dantzig-Wolfe decomposition we refer to Dantzig and Wolfe (1960); Desrosiers and Lübbecke (2005). The ECPP can be Dantzig-Wolfe decomposed in a number of different ways. Decomposition by classes has been found to be most intuitive, i.e. a subproblem is formed for each combinations of a course and a block. This decomposition is done partially, such that the part of the model concerning existing classes is left as-is. A solution to a subproblem is hence a set of students attending the given course in the given block. It will be shown that this decomposition approach leads to subproblems which we are able to solve to optimality with a greedy algorithm.

5.4.1 The Master Problem

It is the job of the master problem to select those columns, generated by the subproblems, which fulfills as many elective course requests as possible, while maintaining the necessary constraints. Each column corresponds to a *number* of classes of a specific course given in a specific block. To enumerate the columns, a new index for each course block pair (c, b), $p \in P_{c,b}$ is used. For each column a binary variable $z^p_{c,b} \in \{0, 1\}$ is representing the usage of the corresponding column. If a student s is assigned to the course c in block b in column p the constant $A^{s,p}_{c,b} \in \{0, 1\}$ is equal to 1 and otherwise zero. The number of classes which is required for the course c in block b in column p is given by the constant $D^p_{c,b} \in Z^+$. With these definitions, the full master model (5.4.1) is created.

Decomposed ECPP - Master Problem (5.4.1)

$$\max \quad \sum_{c,b,s,p} A^{s,p}_{c,b} \cdot z^p_{c,b} \tag{5.4.1a}$$

$$\text{s.t.} \quad \sum_{c,p} A^{s,p}_{c,b} \cdot z^p_{c,b} + \sum_{t} E^{s,t} \cdot u^t_b \leq 1 \qquad \forall\, b, s \tag{5.4.1b}$$

$$\sum_{b,p} A^{s,p}_{c,b} \cdot z^p_{c,b} \leq V^s_c \qquad \forall\, c, s \tag{5.4.1c}$$

$$\sum_{c,b,p} D^p_{c,b} \cdot z^p_{c,b} \leq Q \tag{5.4.1d}$$

$$\sum_{c,p} SC_{c,f} \cdot D^p_{c,b} \cdot z^p_{c,b} + \sum_{c,t} SC_{c,f} \cdot H^t_c \cdot u^t_b \leq SM_f \qquad \forall\, b, f, SM_f > 0 \tag{5.4.1e}$$

$$\sum_{b} u^t_b = 1 \qquad \forall\, t \tag{5.4.1f}$$

$$\sum_{p} z^p_{c,b} \leq 1 \qquad \forall\, c, b \tag{5.4.1g}$$

$$z^p_{c,b} \in \{0, 1\} \tag{5.4.1h}$$

$$u^t_b \in \{0, 1\} \tag{5.4.1i}$$

The objective function of the master problem, like objective function (5.3.1a), calculates the number of satisfied elective course requests. Constraint (5.4.1b) ensures that no student is taught in two courses in the same block. Constraint (5.4.1c) ensures that an elective course request is only satisfied once. Constraint (5.4.1d) ensures that no more than Q classes are established. Constraint (5.4.1e) ensures that the resource limitations of subject f are not exceeded. Constraint (5.4.1f) ensures that the existing classes are taught. Finally Constraint (5.4.1g) ensures that only one non-zero $z^p_{c,b}$ is applied for each pair (c, b). This is the convexity constraint.

The full master problem (5.4.1) is analogous to the direct model (5.3.1). Unfortunately the master problem has an exponential number of variables, hence column generation is applied to solve the relaxed master problem where the $z^p_{c,b}$ variables and the u^t_b variables are relaxed such that $z^p_{c,b} \in [0, 1]$ and $u^t_b \in [0, 1]$. Now it is possible to start with a restricted master problem where only a subset of $z^p_{c,b}$ variables are used and more variables are only added if necessary. The subproblem is described in Section 5.4.2, and is is dependent on the dual variables of the constraints in model (5.4.1), see Table 5.2.

Table 5.2: List of dual variables and the corresponding bounds for the master problem

Constraint	Dual variable	Bound
(5.4.1b)	α^s_b	≥ 0
(5.4.1c)	β^s_c	≥ 0
(5.4.1d)	γ	≥ 0
(5.4.1e)	$\delta_{b,f}$	≥ 0
(5.4.1f)	η^t	free
(5.4.1g)	$\phi_{c,b}$	≥ 0

5.4.2 Subproblem

A subproblem is defined for each combination of a block and a course, i.e. for each (c, b). Given a course c and a block b the subproblem should find the set of students $S' \in S$ who is taught the course c in block b with the maximal reduced profits. Notice that several classes of course c may have to be given to teach the S' students. Given a fixed block b and a fixed course c the binary variable x^s defines whether student s should be included into the set of students S'. The number of classes which are required to teach the S' students is defined by the integer variable y. With these definitions we can now present the entire subproblem in model (5.4.3).

Decomposed ECPP - Subproblem (5.4.3)

$$\max \quad C^{\text{red}}_{c,b} = -\sum_s (\alpha^s + \beta^s - W^s) \cdot x^s - (\gamma + \sum_f \delta_f) \cdot y - \phi \tag{5.4.3a}$$

$$\text{s.t.} \quad x^s \leq V^s \qquad \forall\ s \tag{5.4.3b}$$

$$\sum_s x^s \geq L \cdot y \tag{5.4.3c}$$

$$\sum_s x^s \leq U \cdot y \tag{5.4.3d}$$

$$y \leq Q \tag{5.4.3e}$$

$$SC_f \cdot y \leq SM_f \qquad \forall\ f, SM_f > 0 \tag{5.4.3f}$$

$$x^s \in \{0, 1\} \tag{5.4.3g}$$

$$y \in \mathbb{N}_0 \tag{5.4.3h}$$

The objective function of the subproblem defines the reduced profit for the subproblem, given the current value of the dual variables α^s_b, β^s_c, γ, $\delta_{b,f}$ and $\phi_{c,b}$ from the master problem. Constraint

(5.4.3b) ensures that only students who request the course c can be enrolled into S'. Constraint (5.4.3c) ensures that enough students are enrolled to satisfy the minimal class size and constraint (5.4.3d) ensures that no more students are enrolled than the maximal class size times the number of classes. Constraint (5.4.3e) ensures that the no more than the maximal number of allowed classes are created. Finally constraint (5.4.3f) ensures that the resource limitations are not exceeded.

5.4.3 Combinatorial solution of the Subproblem

It will now be shown that subproblem can be solved to optimality using a greedy algorithm with a time complexity of $O(|S|\log(|S|))$. This is possible due to the uniform knapsack structure of constraint (5.4.3d), i.e. all students take up the same amount of space in each elective course class. Therefore a greedy algorithm can be written by simply processing the students in order of their contribution to the objective. So even though a lot of different subproblems exist, it will be possible to solve these in an small amount of time. Remember that a course c and a block b is given explicitly.

The objective of the subproblem is now divided in two parts. I.e. a part defined for each of the variables x^s and y. Let p^s define the contribution student s makes to the objective if he is included in the solution. It is not feasible to include student s in the solution if the student has not requested course c. This is denoted by a contribution to the objective of $-\infty$.

$$p^s = \begin{cases} -(\alpha^s + \beta^s - W^s) & V_c^s = 1 \\ -\infty & \text{Otherwise} \end{cases} \tag{5.4.4}$$

Likewise r is the contribution to the objective given by an increase of y by one.

$$r = -\gamma - \sum_f \delta_f \tag{5.4.5}$$

By these definitions the objective can be written as

$$C_{c,b}^{\text{red}} = \sum_s p^s x^s + r \cdot y - \phi \tag{5.4.6}$$

Notice that, by Table 5.2, $r \leq 0$, whereas p^s has unrestricted sign. This entails the following observation: The objective of the subproblem is to find those students which maximizes $\sum_s p^s$, while y should be selected as low as possible, i.e. the students should be fitted into as large classes as possible. The maximal number of classes N which can be created for a course c and a block b is given by equation (5.4.7).

$$N = \min\big(Q, SM_{\bar{f}}\big) \tag{5.4.7}$$

Given these definitions we are now ready to present a combinatorial algorithm for finding the optimal set of students S^{opt} which corresponds to the optimal solution of the subproblem, see Algorithm 8. The algorithm gradually builds up a set of students S^{opt} which constitutes an optimal solution. First all the students with positive contributions p^s are included, line 5-7. If the number of student in the class S' is below the lower class limit, extra students are added in line 8-13. Finally it is checked if the group of students in S' will improve the current solution S^{opt}, if

yes S' is included into the solution otherwise the algorithm terminates.

Algorithm 8: Revised Subproblem (c, b)

Input: p^s, r
Output: Optimal set S'

Let $\bar{s}$ denotes the student with highest p^s value, not already included in $S^{opt} \cup S'$
$y = 0, S^{opt} = \{\}$
while $y < N$ **do**
  $S' = \{\}$
  while $|S'| < U$ **and** $p(\bar{s}) > 0$ **do**
    Add $\bar{s}$ to S'
    Update $\bar{s}$
  if $|S'| < L$ **then**
    Calculate $n = |S^{opt}| + |S'| - L \cdot (y + 1)$
    Let $\hat{s}$ denotes the student with highest p^s value, not included in $S^{opt} \cup S'$
    while $n < 0$ **do**
      Add $\hat{s}$ to S'
      Update n and $\hat{s}$
  if $\sum_{s \in S'} p^s + r > 0$ **then**
    $S^{opt} = S^{opt} \cup S'$
    $y = y + 1$
  else
    STOP
if $\sum_{s \in S'} p^s + r - \phi > 0$ **then**
  Add column of S^{opt} to master problem

5.4.4 Solving the ECPP by Column Generation

The Dantzig-Wolfe decomposition of the ECPP, defined by the relaxed master problem from model (5.4.1) and the subproblem model (5.4.3), is solved using the standard Column Generation framework (see e.g. Barnhart et al. (1998)). A simple heuristic is used to find the initial feasible solution. For each course/block pair, create a column which contains all the students which have elected this course. As all constraints of the decomposed part of the problem are set packing constraints, no columns are needed to ensure feasibility.

5.4.5 Explicit Constraint Branching

Explicit Constraint Branching (ECB) is a rather new technique used to improve the performance of Branch and Bound algorithms. The idea is to divide the set of variables into smaller subsets and explicitly add constraints to the problem, and possible also new variables. These additions equip the model with additional structure which may provide superior branching. The approach is generally used when the problem structure required for general constraint branching is lacking. ECB is originally developed for standard MIP solving, see Appleget and Wood (2000).

Suppose a generic MIP is solved, with the integer variable $x_j \geq 0$, $j \in J$. Then let $J' \subseteq J$ be a subset of J and define integer coefficient α_j for each $j \in J'$ such that $\sum_j \alpha_j x_j$ must be integer in any solution to the MIP. A branching scheme to this problem is derived from

$$\sum_{j \in J'} \alpha_j x_j \leq m \qquad \text{or} \qquad \sum_{j \in J'} \alpha_j x_j \geq m + 1 \tag{5.4.9}$$

where m is any integer. This implies the following steps, which constitute ECB.

- Define the subsets of J_k of the index set J of integer variables.
- Define the integer coefficients $\alpha_{k,j}$ for each k and for each $j \in J_k$.
- Define the general integer ECB variables y_k for each k.
- Add the ECB constraints.

$$\sum_{j \in J_k} \alpha_{k,j} x_j - y_k = 0 \qquad \forall\ k \tag{5.4.10}$$

Constraint branching is then performed by standard variable branching on the variable y_k. It should be mentioned that the subsets J_k can even be defined dynamically such that these are updated in each iteration of the branching procedure. The idea is that the variables should be roughly evenly divided such that in each subset there is equally many fractional variables, variables with value zero and variables with value one. Moreover if none of $\sum_{j \in J_k} x_j$ is fractional then one or some of the fractional variables should be moved between the subsets such that every sum is fractional.

Keeping the subsets J_k static throughout the process is known as static ECB. The following is the basic ECB constraint,

$$\sum_{j \in J} x_j - y_j = 0 \tag{5.4.11}$$

which is simply defined over the entire set of variables J. In principle the ECB constraints can be defined over the already existing indices in the model. This approach is applied to the basic ECPP model (5.3.1), by the following steps:

- Define static ECB-constraint by

$$\sum_{b} y_{c,b} - o_c = 0 \qquad \forall c \tag{5.4.12}$$

  where the ECB variable to be branched on is $o_c \in \mathbb{N}_0$. I.e. o_c is the number of classes teaching course c. According to Appleget and Wood (2000) branching over this kind of variable breaks some of the symmetry of the model.

- Solve the model with a standard MIP solver with branching priority for o_c set to highest among all variables.

Using ECB in a Branch and Price Context

ECB can indeed also be used in a Branch and Price framework, where branching is performed on the master problem. Again we choose to apply constraints which are defined over the already existing indices. For instance the following,

$$\underline{\rho}_c \ \leq \ \sum_{b,p} D^p_{c,b} z^p_{c,b} \ \leq \ \overline{\rho}_c, \qquad \forall c \tag{5.4.13}$$

which defines a lower and an upper bound on the total number of classes teaching course c. Initially $\underline{\rho}_c = 0$ and $\overline{\rho}_c = Q$ and branching is performed by changing these bounds. Suppose that for a given c expression (5.4.13) takes the fractional value l. The tree is now split by the two conditions

$$\underline{\rho}_c = \lceil l \rceil \quad \text{and} \quad \overline{\rho}_c = Q \tag{5.4.14}$$

$$\underline{\rho}_c = 0 \quad \text{and} \quad \overline{\rho}_c = \lfloor l \rfloor \tag{5.4.15}$$

which excludes the fractional solution from the feasible area, but maintains all integer solution. Notice that condition (5.4.14) will potentially result in an infeasible MP. Therefore dummy columns should be added such that feasibility is maintained. Another constraint which could be used exactly the same way is the following

$$\sum_{c,p} D^p_{c,b} z^p_{c,b} - r_b = 0, \qquad \forall b \tag{5.4.16}$$

which is the total number of classes being taught in each block. Yet another is

$$\sum_{b,s,p} A^{p,s}_{c,b} z^p_{c,b} - \varrho_c = 0, \qquad \forall c \tag{5.4.17}$$

which is the total number of student being taught course c. A Branch and Price algorithm using ECB will be tested in Section 5.5. Note that even if all ECB variables are integer it is not guaranteed that the solution is in fact integral. Therefore a Branch and Price algorithm using this form of ECB should also implement a second priority branching scheme, such as standard variable branching. Note that imposing ECB constraints results in very little modification of the DWD. The additional constraints are imposed on the MP, which results in new dual variables. However it is trivial to incorporate this new dual variable in the revised subproblem. E.g. the dual variable of constraint 5.4.13 is simply added to equation (5.4.5).

5.5 Results

In the previous sections we have suggested four different algorithms. In the initial tests the Branch-and-Price algorithm without the ECB constraints had inferior performance and will hence not be further tested. The remaining three optimization algorithms which we will test are:

- Solving the basic model (5.3.1) using a standard MIP solver.
- Solving the basic model (5.3.1) with both the basic ECB constraint (5.4.11) and constraint (5.4.12), using a standard MIP solver.
- Branch and Price algorithm using equations (5.4.13) and (5.4.16) in an ECB branching scheme, as described in Section 5.4.5.

The three algorithms are tested on datasets from 98 Danish high schools for the year 2008. The current version of the high school administration system Lectio applies the meta-heuristic developed by Binzer and Kjeldsen (2008). Given that the three suggested algorithms can achieve optimal results, it is now possible to evaluate the efficiency of the meta-heuristic in Lectio. Tests are ran on a notebook equipped with an Intel Core2 T7200 CPU @ 2 GHz, 4 GB of RAM, and running Windows Vista 32bit. This particular CPU has 2 cores, which is irrelevant as no parallelization has been implemented. The BnP algorithms have been implemented in Microsoft Visual C# using .NET framework version 3.5. The direct models have been implemented in GAMS 22.6.149. For all tests CPLEX 10.0 has been used as solver.

The number of elective course requests which can be fulfilled depends upon the choice of number of blocks and the choice of number of classes. These quantities are attempted selected such that they both are somewhat binding. In Binzer and Kjeldsen (2008) an inspection of dual bounds with respect to number of elective classes is performed. The conclusion is that increasing the number of elective course classes allows for granting more course requests, which is somewhat obvious. However if the number of classes is selected very high, the elective course planning reduces into the more simple matter of assigning students to blocks. A similar analysis could be made for the choice of number of blocks. This motivates the following choice of the maximum number of elective course classes available, see equation (5.5.1).

$$Q = \left\lceil \sum_c \left(\frac{\sum_s V_c^s}{U_c} \right) \right\rceil \tag{5.5.1}$$

A choice on the number of blocks should also be made. The reason why $|B|$ is not predefined is because the high schools usually select this quantity using an ad-hoc procedure. The high school administration will usually like to see solutions for several values of $|B|$. In the following we attempt to derive a formula which selects a tight value for $|B|$. A tight value is preferred, as it provides more interesting results. Let k_1 denote the highest number of existing classes which a single student is attending,

$$k_1 = \max_s \sum_t E^{t,s} \tag{5.5.2}$$

At least k_1 blocks should always be established to ensure feasibility. Let k_2 denote the rounded average number of courses pr. student,

$$k_2 = \text{Round}\left(\frac{\sum_{c,s,t}(V_c^s + E^{s,t})}{|S|} \right) \tag{5.5.3}$$

By inspection of the data, selecting the number of blocks equal to k_2 yields an extremely tight problem. Therefore it seems appropriate to always select a slightly higher value than this. We have chosen to select the number of blocks using equation (5.5.4), such that the number of blocks can never be lower than $k_2 + 1$.

$$|B| = \max(k_1, k_2 + 1) \tag{5.5.4}$$

5.5.1 Performance

Table 5.5 contains the average running time and the gap for all problems. It should be noted that for those problems not solved within one hour, $3600s$ is used for the calculation of the average time. The table shows that in average Direct/w ECB performs best and BnP/w ECB performs worst. Furthermore it is seen that Direct/w ECB in no cases provided a gap worse than 5%, which is quite low. The BnP/w ECB generally performs bad, and even worse than the pure direct model, which is disappointing.

Table 5.5: Running time and gaps for the algorithms

	Direct	Direct/w ECB	BnP/w ECB
Average time	$1373s$	$1121s$	$1523s$
Average gap	0.9%	0.6%	2.8%*
Max gap	7.0%	5.0%	29.0%

* Three problems did not result in a gap within one hour

Table 5.6 contains the percentage of problems solved to optimality within different time spans. The number of solved problems are indicated in brackets.

The table also shows that Direct/w ECB clearly performs best, solving 50% of the data instances in less than a minute.

Table 5.8 shows a comparison between the meta-heuristic which is currently applied to the ECPP and the direct method with ECB from this paper. It is seen that the exact method does provide a improvement. Even though the average difference is small it must be considered interesting for the high schools to get access to a better approach. For some schools an improvement of 10 classes could be found, and for these high schools the accessibility for a better solution must

Table 5.6: Percentage of solved problems

Time(s)	Direct	Direct /w ECB	BnP /w ECB
¡ 60	45.9 % (45)	50.0 % (49)	35.7 % (35)
60 - 600	12.2 % (12)	12.2 % (12)	19.4 % (19)
600 - 1800	6.1 % (6)	8.2 % (8)	3.1 % (4)
1800 - 3600	1.0 % (1)	3.1 % (3)	1.0 % (1)
¿ 3600*	34.7 % (34)	26.5 % (26)	31.6 % (39)

* Problems not solved to optimality within one hour

Table 5.7: Results for a given set of real-life problems at Danish high schools.

	No.of student	No.of requests	No.of courses	No. blocks	Objective	Assigned classes	Assigned requests
Vejen	382	586	29	3	69572	36	586
Silkeborg	927	1789	65	5	208203	77	1786
Falkoner	421	1080	49	4	127690	66	1080
Vordingborg	415	1462	61	5	182790	68	1462
Alssund	385	650	31	5	64271	34	645
Holstebro	345	567	18	5	56700	29	567
Frederikssund	159	273	18	4	28690	18	273

Table 5.8: Comparison between Direct /w ECB and the Meta-heuristic solver.

	Direct /w ECB	Meta-heuristic	Abs. diff	Rel. diff
Average	27	28	1.1	2.9

be considered very important. It should be noted that the running time for the Meta-heuristic is roughly 2 minutes whereas the running time for the Direct method with ECB is one hour. However in empirical experiments performed in Kristiansen and Sørensen (2010) it is shown that for a small sample of datasets, Direct /w ECB also performs better with a running time of 2 minutes. In the majority of cases, a running time of 2 minutes actually provided the same best solution as with a running time of 1 hour.

5.5.2 Extension

It is preferable if elective courses classes consist of students from the same common classes, as this entails several benefits from a social point of view. This is done by extending the ECPP such that it minimizes the total number of represented common classes while simultaneously maximizing the number of granted elective course requests. The extension is added using the ϵ-constraint method (Ehrgott (2000)), such that the problem is multi-objective. Given the result of the Direct/w ECB algorithm, an extended model is formulated, using a new binary variable $v^k_{c,b} \in \{0,1\}$. This variable states whether any student from common class k is taught course c in block b. The objective for the new model is then simply to minimize the total sum of common classes, see equation (5.5.9).

$$\min \sum_{c} \sum_{b} \sum_{k} v^k_{c,b} \tag{5.5.9}$$

Instead of the old objective function (5.3.1a), a new constraint on the number of granted elective class requests is included, see equation (5.5.10).

$$\sum_{c}\sum_{b}\sum_{s} W_c^s \cdot x_{c,b}^s \geq \epsilon \tag{5.5.10}$$

Finally a link between the $v_{c,b}^k$ variables and the $x_{c,b}^s$ variables is given, see equation (5.5.11). The matrix $G^{s,k}$ is an incidence matrix which is 1 if student s is part of common class k and 0 otherwise. The following constraint ensures that if one common class is represented in a course in a block, it is counted.

$$v_{c,b}^k \geq x_{c,b}^s \quad \forall\, c, b, s, k, G^{s,k} = 1 \tag{5.5.11}$$

The parameter ϵ defines the acceptable number of granted elective courses, compared to the optimal solution of direct model, see equation (5.5.12).

$$\epsilon = R \cdot \sum_{c,b,s} \bar{x}_{c,b}^s \tag{5.5.12}$$

By adjusting the ratio R to different values we generate the results in Table 5.13. The closer to 1 the diverge percentage R is the less the extended solution differs from the original solution. The table lists the objective solutions. The total number of represented common classes is shown in brackets. Basic denotes the solution from the model without use of the extension.

Table 5.13: Performance test of common class extension

	Objective					
ID	Basic	$R = 1.00$	$R = 0.95$	$R = 0.90$	$R = 0.75$	$R = 0.50$
Avedøre	347 (54)	347 (39)	330 (32)	313 (29)	263 (21)	174 (12)
Esbjerg G.	417 (85)	417 (46)	397 (36)*	379 (33)*	313 (24)	209 (14)
Metropolitan	419 (65)	419 (52)*	399 (42)*	378 (37)*	315 (28)	213 (16)
Nordsjællands	155 (24)	155 (20)	148 (16)	140 (15)	117 (11)	78 (6)
Nærum	972 (190)	972 (115)	924 (89)	875 (76)	729 (52)	486 (29)
Rysensteen	313 (31)	313 (21)*	298 (19)*	282 (17)	237 (14)	159 (8)
Silkeborg	669 (195)	669 (111)*	636 (85)*	604 (72)	502 (46)	339 (22)
Skanderborg	384 (47)	384 (39)	365 (24)	346 (20)	288 (13)	201 (8)
Stenhus	1056(229)	1056 (144)*	1004 (109)*	951 (92)	792 (63)	528 (33)
Støvring	334 (58)	334 (36)	318 (31)	301 (27)	251 (19)	167 (11)
Aalborg	880 (182)	880 (139)*	836 (103)*	792 (88)	660 (60)	440 (33)
Aalborghus	370 (75)	370 (49)	352 (38)	334 (33)	280 (24)	189 (13)
Average	526 (103)	526 (68)	501 (52)	475 (45)	396 (31)	265 (17)

* Not solved to optimality within one hour

It is clearly seen that if the ratio R is low, fewer elective course requests are granted and less common classes are represented. The most interesting result from this test are the difference between Basic and $R = 1.0$. The number of granted elective course requests is the same, but there is a big difference in the number of represented common classes. On average 103 common classes are represented using the basic ECPP whereas with a diverge percentage of 1 only 68 common classes are used. An improvement of 34%. For some of the larger data sets the improvement are more than 40%. The reason that the extended model is able to improve the solution so significantly is due to symmetry. As mentioned previously the ECPP contains a great deal of symmetry. The extension takes advantage of this to swap students such that the number of represented common classes are minimized while the number of granted requests remains high.

5.6 Conclusion

In this article we have demonstrated how a critical planning problem for the Danish high schools can be optimized by applying three different approaches; Direct MIP approach, MIP with Explicit Constraint Branching and Branch-and-Price with Explicit Constraint Branching. The algorithms are tested on 98 data sets from different high schools of school-year 2008. The tests show that Explicit Constraint Branching is an interesting tool. Furthermore, the tests reveal that the current meta-heuristic algorithm can be significantly improved. Furthermore, we have shown that an important secondary objective, minimization of the number of common classes, can improve the solution substantially using the ϵ-constraint method. Finally it should be mentioned that currently the approaches in this article are not implemented in any software tools accessible for the Danish high schools. However, showing that the elective course planning can be improved has indeed raised interest among several high schools.

Bibliography

J. Appleget and R. Wood. *Explicit-Constraint Branching for Solving Mixed-Integer Programs*, chapter 14, pages 245–262. Springer Netherlands, 2000.

C. Barnhart, E. Johnson, G. Nemhauser, M. Savelsbergh, and P. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, March 1998. ISSN 0030-364X.

S. P. Binzer and S. H. Kjeldsen. Metaheuristics for high school planning. Master's thesis, IMM, DTU, 2008.

D. Costa. A tabu search algorithm for computing an operational timetable. *European Journal of Operational Research*, 76(1):98 – 110, 1994. ISSN 0377-2217.

G. Dantzig and P. Wolfe. Decomposition principle for linear programs. *Operations Research*, 8 (1):101–111, 1960.

D. de Werra. An introduction to timetabling. *European Journal of Operational Research*, 19(2): 151 – 162, 1985. ISSN 0377-2217.

J. Desrosiers and M. Lübbecke. Selected topics in column generation. *G–2002–64*, 34:1–34, 2005.

M. Ehrgott. *Multicriteria Optimization*. Springer, 2000.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

V. Kaibel, M. Peinhardt, and M. Pfetsch. Orbitopal fixing. In M. Fischetti and D. Williamson, editors, *Integer Programming and Combinatorial Optimization*, volume 4513 of *Lecture Notes in Computer Science*, pages 74–88. Springer Berlin / Heidelberg, 2007.

S. Kristiansen and M. Sørensen. The class packing problem. Master's thesis, DTU-Management, 2010.

G. Laporte and S. Desroches. The problem of assigning students to course sections in a large engineering school. *Computers & Operations Research*, 13(4):387 – 394, 1986. ISSN 0305-0548.

R. Lewis, B. Paechter, and B. McCollum. Post enrolment based course timetabling: A description of the problem model used for track two of the second international timetabling competition. Cardiff Accounting and Finance Working Papers A2007/3, Cardiff University, Cardiff Business School, Accounting and Finance Section, 2007.

F. Margot. Exploiting orbits in symmetric ilp. *Mathematical Programming*, 98:3–21, 2003. ISSN 0025-5610.

T. Müller. Itc2007 solver description: a hybrid approach. *Annals of Operations Research*, 172: 429–446, 2009. ISSN 0254-5330.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

T. Müller, K. Murray, and S. Schluttenhofer. University course timetabling & student sectioning system, 2007. Space Management and Academic Scheduling, Purdue University.

H. Rudova and K. Murray. University course timetabling with soft constraints. In *Practice And Theory of Automated Timetabling IV.*, pages 310–328, 2003.

A. Tripathy. School timetabling–a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, 1984.

F. Vanderbeck. On dantzig-wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. *Operations Research*, 48:111–128, January 2000. ISSN 0030-364X.

Chapter 6

Elective Course Student Sectioning at Danish High Schools

Simon Kristiansen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, thst@dtu.dk

† MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

Abstract The *Elective Course Student Sectioning* (ECSS) problem is a yearly recurrent planning problem at the Danish high schools. The problem is of assigning students to elective classes given their requests such that as many requests are fulfilled and the violation of the soft constraints is minimized. This paper presents an adaptive large neighborhood search heuristic for the ESCC. The algorithm is applied to 80 real-life instances from Danish high schools and compared with solutions found by using the state-of-the-art MIP solver Gurobi. The algorithm has been implemented in the commercial product Lectio, and is thereby available for approximately 200 high schools in Denmark.

Keywords: Education Timetabling; High School Timetabling; Student Sectioning; Elective Course Planning; Adaptive Large Neighborhood Search; Integer Programming

6.1 Introduction

The purpose of this article is to describe the *Elective Course Student Sectioning* (ECSS) problem and the solution methods used to solve it. ECSS serves as a pre-processing planning problem for the actual high school timetabling in Denmark and is an important yearly recurrent planning problem. The problem is concerned with assigning students to elective classes given their requests, subject to various soft and hard constraints.

This paper is written in collaboration with the Danish software company MaCom A/S, whose main product is the cloud-based high school administration system Lectio. Lectio is used by the majority of all the Danish high schools. Due to the diversity of high schools, it is very important that the problem is formulated such that it covers all the high schools and such that the solution approach performs well for problems of different sizes and format.

[1] Accepted for publication in the special issue of *Annals of Operations Research* in collaboration with *PATAT2012 - 9th International Conference on the Practice and Theory of Automated Timetabling. (2013)*

In Section 6.2 the ECSS is described in detail with a literature review given in Section 6.3. In Section 6.4 the problem is formulated as a Mixed Integer Programming problem and the complexity of the problem is proven. The solution approach for the problem is explained in Section 6.5. Parameter tuning and the performance results are presented in Section 6.7 and 6.8, respectively. Finally Section 6.9 rounds up and concludes.

6.2 Problem Description

It takes three years to complete a high school education in Denmark and every year the high schools need to create a timetable for the following year, hence they also need to solve the ECSS. Each year the students request some elective courses which they want to attend along with their mandatory courses. The problem is then to assign the students to elective classes given their requests and assign these classes to some time slots. The main goal of the ECSS is to fulfill as many of the students' elective course requests as possible while minimizing the number of classes created. The problem is of both educational and economical nature. First of all, the students are planning their high school education such that they have the necessary merits for applying for a university education afterwards. If a student is not granted his requests he might miss a merit to get admission to his desired education. Secondly if a student is not granted his request, it might entail that the student changes high school or drops out, and this imposes some economic issues for the high school. The Danish high schools are self-governing and get a fee from the state based on the number of students which finish an education at the high school, i.e. a significant part of the high schools income is based on the students. It should be noted that the students in Denmark can freely choose between high schools. Another aspect of the economic issues is the creation of classes. For each created class there is a cost of approximately 27.000 € p.a., i.e. it is not enough to grant all the requests it is also very important for the high schools to keep the number of created classes at a minimum.

A typically weekly schedule for a Danish high school consists of five days, each day divided into a given number of time slots where the teaching is performed. Each time slot then consists of a number of lectures and a student can of course only attend one course in each time slot. There exist two types of time slots in the Danish high schools. Time slots for the mandatory courses and time slots for the elective courses. This is due to the mandatory courses often being taught in cohorts of students, i.e. a student of a given cohort has almost all his mandatory courses with the fellow students from the same cohort. The elective courses are however mixed between these cohorts, and hence it is more beneficial for the administration to have the elective classes in some separate time slots. Figure 6.1 is an example of a typical weekly schedule containing five different time slots for elective courses.

	Monday	Tuesday	Wednesday	Thursday	Friday
8:15 9:45	Time1			Time3	
10:00 11:30					
	Lunch break				
12:00 13:30					
13:45 15:15		Time2		Time4	Time5

Figure 6.1: An example of a weekly schedule with four time slots each day. Five time slots (gray colored) are reserved for the elective courses whereas the mandatory courses are placed in remaining time slots (white colored)

The time slots for elective courses are often chosen to be placed in the beginning or in the end of

a day. This is to minimize the possibility of creating idle slots for students when creating the entire timetable. Due to the dividing of the time slots into elective and mandatory, the elective course planning can be seen as an independent part of the weekly schedule and therefore the mandatory courses and their respective time slots are neglected in the remainder of this article. The problem of assigning the mandatory courses is known as the classical High School Timetabling. (See e.g. Sørensen et al. (2012) and Post et al. (2012))

A student has the possibility to make up to five first priority requests and declare some second priorities if one of the first is not granted. Ideally all the students should be assigned their first priorities for elective course requests. If a student is not granted his first priority requests, it would be preferable if his second priority request is granted. However it should be a noted that the outcome of the ECSS is an algorithm used in decision support software. If a student is not granted one of his first priority requests, the high school administration wants to have a dialog with the given student before assigning him to one of his second priorities. Hence we do only consider the first priority requests of the students.

6.3 Related Literatures

As mentioned the ECSS serves as a pre-processing planning problem for High School Timetabling (HSTT). The last couple of years more research have been done within HSTT and the *third International Timetabling Competition* (ITC2011) treats the HSTT (see e.g. Post et al. (2012) and Sørensen et al. (2012)). The ECSS is however not a very well researched area. Compared to the problem of ITC2011, the ECSS is the matter of which students should be in which elective class, whereas the high school timetabling is concerned with the construction of a timetable according to some predefined classes. Many articles have been made within education timetabling and within this research area ECSS is most related to *Student Sectioning* (Carter and Laporte (1998); Schaerf (1999); Burke and Petrovic (2002); Pillay (2010)).

The literature on Student Sectioning is however mostly concerned the universities, e.g. Erben and Keppler (1996); Rudova and Murray (2003); Müller and Murray (2010). In Müller and Murray (2010) University Course Timetabling and Student Sectioning are combined and solved using the Iterative Forward Search algorithm. Carter (2001) describes *demand-driven timetabling* where student selections of courses are utilized to create a timetable that satisfies as many requests as possible. In de Haan et al. (2007) *optional subjects* for the students are used when constructing the high school timetabling in the Netherlands, and *cluster schemes* are created to maintain the students' optional courses. The optional subjects are similar to the elective courses of this paper.

Due to reform changes in the Danish education sector in 2007, the high schools in Denmark became aware of the need of sufficient solution tools for their planning problems, hence also the ECSS. The ECSS for the Danish high schools was first described in Kristiansen et al. (2011). The article gives a good overview of the problem, and Dantzig-Wolfe decomposition and explicit constraint branching is used for solving the problem. The approach however was only created to clarify the performance of an earlier solution approach used in the software Lectio, and was hence never released and the model misses some of the restriction which is incorporated in this article. It did however prove that the previous solution method for the ECSS was inefficient and that it lacked some restrictions including the fairness distribution which is described in the following section.

6.4 Integer Programming Model

In the following the ECSS is formulated as a MIP model. The problem is a maximization problem which aims at maximizing the number of granted elective course requests while minimizing the violation of soft constraints and respecting the hard constraints. For the ECSS the high schools have a set of students $\mathcal{S}$, a set of offered courses $\mathcal{E}$, a set of classes $\mathcal{C}$ and a set of time slots $\mathcal{T}$. The parameter $D_{c,e} \in \{0,1\}$ denotes whether elective class c is teaching course e and parameter $R_{e,s}$

$\in \{0,1\}$ indicates whether student s has requested elective course e or not. The decision whether student s is assigned elective class c in time slot t is defined by the binary variable $x_{s,c,t} \in \{0,1\}$, whereas the binary variable $y_{c,t} \in \{0,1\}$ takes value 1 if elective class c is assigned time slot t, zero otherwise. ECSS is a timetabling problem and it belongs to the class of NP-hard problems Welsh and Powell (1967). In the following the MIP formulation is divided into small sections.

6.4.1 Availabilities

It is not allowed to assign a student to a class of a course he has not requested, and it is obviously not possible to assign a student to more than one elective class in each time slot. Neither is it possible to assign a elective class to more than one time slot. The following constraints make sure that these restrictions are maintained.

$$\sum_{c,t} D_{c,e} \cdot x_{s,c,t} \leq R_{e,s} \qquad \forall\, e, s \tag{6.4.1}$$

$$\sum_{c} x_{s,c,t} \leq 1 \qquad \forall\, t, s \tag{6.4.2}$$

$$\sum_{t} y_{c,t} \leq 1 \qquad \forall\, c \tag{6.4.3}$$

As the elective courses can have duration of more than one year some of the elective courses may be a continuation from the previous school year. For these elective classes the students are locked, i.e. the students which were assigned the elective class the previous year must be assigned the class this year also. Let $A_{c,s} \in \{0,1\}$ take value 1 if student s is locked to elective class c, zero otherwise. The following constraints are then imposed.

$$x_{s,c,t} \leq y_{c,t} \qquad \forall\, c, t, s, A_{c,s} = 0 \tag{6.4.4}$$

$$x_{s,c,t} = y_{c,t} \qquad \forall\, c, t, s, A_{c,s} = 1 \tag{6.4.5}$$

6.4.2 Resource Limitations

There exists some resource limitation when solving the ECSS. Firstly, it is determined by the Danish educational legislation, that the class size in high schools may not exceed 28 students. This is to make sure that the students have the best possibly environment. However some elective classes might have an even more restricted upper limit. For classes where the students are locked, the upper class size is equal to the number of locked students, such that no new students can be assigned to the given class. Let the parameter $U_c \in \mathbb{N}$ denote the upper class size for elective class c.

$$\sum_{s} x_{s,c,t} \leq U_c \qquad \forall\, c, t \tag{6.4.6}$$

Furthermore, as the price for creating an elective class is approximately 27.000 € p.a., a high school often has an upper limit on how many classes they can afford to create each year. Let $P \in \mathbb{N}$ be the maximum number of classes which can be created in total.

$$\sum_{c,t} y_{c,t} \leq P \tag{6.4.7}$$

The limitation of classes which can be created of a given course is given by the set of classes $\mathcal{C}$ and the parameter $D_{c,e}$.

There also exists some resource limitation on the number of elective classes with the same subject which can be taught in the same time slot. Let $\mathcal{F}$ be the set of course subjects of a high school. Each course is teaching a subjects, such as *physics* or *chemistry*, given by $K_{c,f} \in \{0,1\}$. Due to the limited resources on e.g. rooms and equipment, it is often not possible to assign all

classes of a course to the same given time slot, e.g. if a high schools only have x physic class rooms, it is not possible to assign more than x physic classes to a given time slot. Let $B_{f,t} \in \mathbb{N}$ be the maximum number of elective classes of subject f which can be taught in time slot t. This imposes the following constraints

$$\sum_c K_{c,f} \cdot y_{c,t} \leq B_{f,t} \qquad \forall\, f, t, B_{f,t} > 0 \tag{6.4.8}$$

6.4.3 Class Positions

Some elective classes cannot be assigned to the same time slot, e.g. two courses with the same preassigned teacher are not allowed to share the same time slot. Let the parameter $J_{c,c'} \in \{0,1\}$ take value 1 if the elective classes c and c' cannot be in the same time slot, zero otherwise. The constraints are given by

$$y_{c,t} + y_{c',t} \leq 1 \qquad \forall\, c, c', t, J_{c,c'} = 1 \tag{6.4.9}$$

On the contrary some courses are forced to be placed in the same time slot. Let $L_{e,e'} \in \{0,1\}$ denote whether classes of course e should be assigned the same time as classes of course e'. As not all elective classes are being assigned to a time slot, a new variable is introduced such that only the assigned classes are considered for this constraint. Let the binary variable $h_{e,t} \in \{0,1\}$, take value 1 if course e is placed in time slot t. We then get the following constraints.

$$y_{c,t} \leq D_{c,e} \cdot h_{e,t} \qquad \forall\, c, e, t \tag{6.4.10}$$

$$h_{e,t} = h_{e,t'} \qquad \forall\, e, e', t, L_{e,e'} = 1 \tag{6.4.11}$$

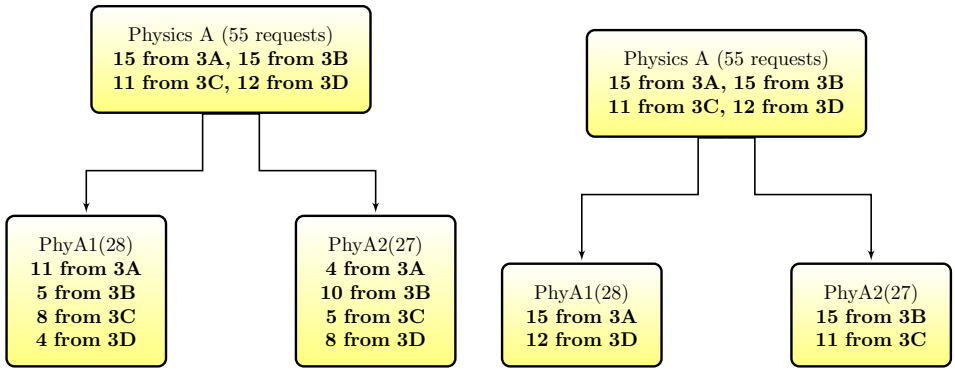
$$\sum_e h_{e,t} \leq 1 \qquad \forall, e, e', t, L_{e,e'} = 1 \tag{6.4.12}$$

6.4.4 Cohorts

When a student is enrolled at a high school he is assigned to a cohort. Many of the mandatory courses in the Danish high schools are taught in classes exactly equal to a cohort. It has the advantage that the students in a cohort are quite familiar with each other, and it makes it easier to collaborate between mandatory classes of different subject, as the students attending the two classes are the same. Hence it would also be beneficial to have as few cohorts representing in each elective class. As the electives can be selected by all the students it is most unlikely that only students from one cohort have requested a given elective. Figure 6.13 gives an example of the handling of cohorts.

We want to minimize the number of cohorts represented in each created elective class. Let $\mathcal{Q}$ be the set of cohorts for a high school, and let $I_{s,q} \in \{0,1\}$ denotes whether student s is part of cohort q or not. The decision variable $z_{c,q} \in \{0,1\}$ indicates whether cohort q is represented in elective class c, or not. There is no need to minimize the number of cohort represented in classes where the students are locked, as these cannot be improved. Let parameter $A_c \in \{0,1\}$ denotes whether elective class c is locked or not. This gives the following constraints.

$$\sum_t I_{s,q} \cdot x_{s,c,t} \leq z_{c,q} \qquad \forall\, c, q, s, A_c = 0 \tag{6.4.14}$$



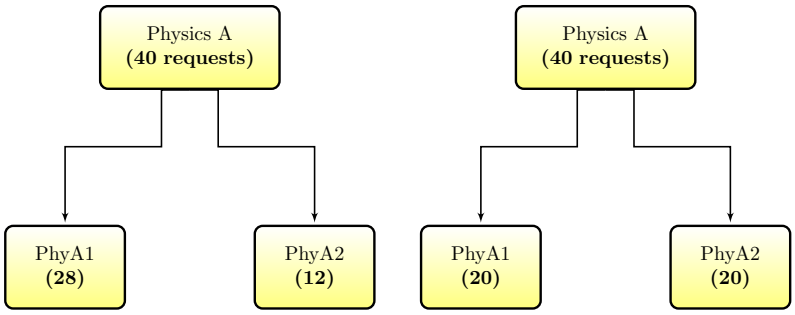
(a) Two classes with four cohorts represented in each class. All possible cohorts are represented in both class.

(b) Two classes with only two cohorts represented in each. Neither of the cohorts is represented in more than one class.

Figure 6.13: Two examples of representing cohorts in classes of same course. 55 students from four different cohorts have requested *Physics A* as elective course. With an upper class size of 28, two classes are needed to fulfill all requests. Figure 6.13(a) is the worst possible scenario where all possible cohorts are represented in each class, whereas 6.13(b) is the best solution where only two cohorts are represented in each class, i.e. no cohort is represented more than once.

6.4.5 Even Distribution

When having two classes of same course, it is then highly appreciated to have approximately the same amount of students attending both classes. This is due to fairness of both the students and the teachers. See Figure 6.15 for an illustration of two different distributions of students in two classes of same course.



(a) The worst distribution

(b) The best distribution

Figure 6.15: Two examples of distribution of students in two classes of same course. 40 students have requested *Physics A* as elective course and there is an upper class size of 28, i.e. two classes are needed to fulfill all requests. Figure 6.15(a) is the worst possible scenario on distribution the students into the classes, whereas 6.15(b) is the best.

We want to minimize the difference between the numbers of students in classes of same course. Only unlocked assigned classes are considered. Let variable $w_{c,c'} \in [0,1]$ be the ratio of the difference between c and c', and let $g_{c,c'} \in [0,1]$ be a slack variable helping with determination of the difference ratio. Let $\rho(i)$ denote the ordinal number of i. The following constraints are imposed

$$\sum_{t,s} x_{s,c,t} - \sum_{t,s} x_{s,c',t} \quad \leq \quad g_{c,c'} \cdot U_c \qquad \begin{array}{l} \forall\, c, c', D_{c,e} = D_{c',e} = 1, \\ A_c = A_{c'} = 0, \rho(c) < \rho(c') \end{array} \tag{6.4.16}$$

$$\sum_{t,s} x_{s,c',t} - \sum_{t,s} x_{s,c,t} \leq g_{c,c'} \cdot U_c \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, A_c = A_{c'} = 0, \rho(c) < \rho(c') \tag{6.4.17}$$

$$\sum_{t} (y_{c,t} + y_{c',t}) - 2 + g_{c,c'} \leq w_{c,c'} \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, A_c = A_{c'} = 0, \rho(c) < \rho(c') \tag{6.4.18}$$

(6.4.19)

6.4.6 Objective Function

ECSS is a maximization problem with a four layered objective. Firstly, and most important, we want to maximize the number of granted requests. The three other parts are minimization parts of the objective. We want to minimize the number of created elective classes and the number of cohorts represented in each elective class. And we want to minimize the difference between the numbers of students in classes of same course.

$$\sum_{c,t,s} \alpha_{c,s} \cdot x_{s,c,t} - \sum_{c,t} \beta_c \cdot y_{c,t} - \gamma \cdot \sum_{c,q} z_{c,q} - \delta \cdot \sum_{c,c'} U_c \cdot w_{c,c'} \tag{6.4.20}$$

The weights vary with courses and students. The definition of the weights are given in Section 6.6.

The full MIP model of the ECSS is given in (6.4.21)

6.4.7 Full MIP Model for ECSS

$$\max \quad \sum_{c,t,s} \alpha_{c,s} \cdot x_{s,c,t} - \sum_{c,t} \beta_c \cdot y_{c,t} - \gamma \cdot \sum_{c,q} z_{c,q} - \delta \cdot \sum_{c,c'} U_c \cdot w_{c,c'} \tag{6.4.21}$$

s.t.

$$\sum_{c} x_{s,c,t} \leq 1 \quad \forall\, t, s$$

$$\sum_{c,t} D_{c,e} \cdot x_{s,c,t} \leq R_{e,s} \quad \forall\, e, s$$

$$\sum_{t} y_{c,t} \leq 1 \quad \forall\, c$$

$$\sum_{s} x_{s,c,t} \leq U_c \quad \forall\, c, t$$

$$\sum_{c,t} y_{c,t} \leq P$$

$$x_{s,c,t} \leq y_{c,t} \quad \forall\, c, t, s, A_{c,s} = 0$$

$$x_{s,c,t} = y_{c,t} \quad \forall\, c, t, s, A_{c,s} = 1$$

$$\sum_{c} K_{c,f} \cdot y_{c,t} \leq B_{f,t} \quad \forall\, f, t, B_{f,t} > 0$$

$$y_{c,t} + y_{c',t} \leq 1 \quad \forall\, c, c', t, J_{c,c'} = 1$$

$$y_{c,t} \leq D_{c,e} \cdot h_{e,t} \quad \forall\, c, e, t$$

$$h_{e,t} = h_{e,t'} \quad \forall\, e, e', t, L_{e,e'} = 1$$

$$\sum_{e} h_{e,t} \leq 1 \quad \forall, e, e', t, L_{e,e'} = 1$$

$$\sum_{t} I_{s,q} \cdot x_{s,c,t} \leq z_{c,q} \quad \forall\, c, q, s, A_c = 0$$

$$\sum_{t,s} x_{s,c,t} - \sum_{t,s} x_{s,c',t} \leq g_{c,c'} \cdot U_c \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, A_c = A_{c'} = 0, \rho(c) < \rho(c')$$

$$\sum_{t,s} x_{s,c',t} - \sum_{t,s} x_{s,c,t} \leq g_{c,c'} \cdot U_c \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, A_c = A_{c'} = 0, \rho(c) < \rho(c')$$

$$\sum_{t} (y_{c,t} + y_{c',t}) - 2 + g_{c,c'} \leq w_{c,c'} \quad \forall\, c, c', D_{c,e} = D_{c',e} = 1, A_c = A_{c'} = 0, \rho(c) < \rho(c')$$

$$x_{s,c,t} \in \{0, 1\}$$

$$y_{c,t} \in \{0, 1\}$$

$z_{c,q} \in \{0,1\}$
$w_{c,c'} \in [0,1]$
$g_{c,c'} \in [0,1]$
$h_{e,t} \in \{0,1\}$

6.5 Solution Methods

It has been chosen to use *Adaptive Large Neighborhood Search* (ALNS) for the ECSS. ALNS is a hyperheuristic first described by Pisinger and Ropke (2005) and is an extension of Large Neighborhood Search (LNS) (Shaw, 1998). Where most neighborhood search algorithms explicitly define the neighborhood, LNS defines the neighborhood implicitly by a removal and an insertion method. ALNS consists of several removal and insertion heuristics. The ALNS framework has the advantage of having many different neighborhoods, such that the algorithm can be able to explore a large part of the solution space, depending on the construction of the heuristics (Pisinger and Ropke, 2005, section 4.3). The pseudo code for the ALNS algorithm as it is presented in Pisinger and Ropke (2010) is shown in Algorithm 1.

Algorithm 1: Adaptive Large Neighborhood Search

Input: a feasible solution $x_{s,c,t}$
1 solution $x^{best} = x$; $\pi = (1, \ldots, 1)$
2 **repeat**
3 select a removal $d \in \Omega^-$ and an insertion heuristic $r \in \Omega^+$ using π
4 $x' = r(d(x))$
5 **if** $c(x') > c(x^{best})$ **then**
6 $x^{best} = x'$
7 **if** *accept(x', x)* **then**
8 $x = x'$
9 update π
10 **until** *stop-criterion met*
11 **return** x^{best}

The set of removal and insertion heuristics are denoted Ω^- and Ω^+, respectively. The variable $\pi \in \mathbb{R}$ which stores the weight of each removal and insertion heuristic is introduced in line 1. Line 1 checks whether the new solution is better than the best known solution. $c(x)$ denotes the objective value of solution x.

In line 1 the solution is evaluated using an accept function. Finally, in line 1 the weights are adjusted based on the performance of each removal and insertion heuristic.

ALNS has been used with success on various problems, especially in *Vehicle Routing Problems*; see e.g. Ropke and Pisinger (2006); Laporte et al. (2010); Azi et al. (2010); Ribeiro and Laporte (2012); Lei et al. (2011). Other problems such as *Lot-Sizing* (Muller and Spoorendonk (2010)), *Resource Constraint Project Scheduling* (Muller (2009)), *Home Health Care Problem* (Steeg and Schröder (2008)). Within educational timetabling problems ALNS has been applied with success for *Consultation Scheduling Problem* (Kristiansen et al. (2013)) and *High School Timetabling* (Sørensen et al. (2012), Sørensen and Stidsen (2013)).

6.5.1 Algorithm Setup

In the following we describe the main elements of the ALNS used for the ECSS. The user-controlled parameters for the ALNS are tuned in Section 6.7.

Adaptive search strategy: The choice of the removal and insertion heuristic is governed by a scoring scheme where each heuristic is assigned a weight which is updated due to its past behavior.

The search is divided into segments of N_{it} consecutive iterations. Let π_h^i be the measure of the performance of heuristic h in segment i. In the first segment all heuristics are assigned the same weight, $\pi_h^{i_0} = 1$. The probability of choosing heuristic h in segment i is given by $\frac{\pi_{h'}^i}{\sum_h \pi_h^i}$. After N_{it} iterations the weights are adjusted according to the score obtained during the segment

$$\pi_h^{i+1} = \eta \cdot \frac{\bar{\pi}_h^i}{a_h^i} + (1-\eta)\pi_h^i \tag{6.5.2}$$

where $\eta \in [0,1]$ is the *reaction factor* and $\bar{\pi}_h^i$ is the number of times heuristic h has been used in segment i. $\bar{\pi}_h^i$ is the observed weight of the heuristic h in segment i and this is updated in each iteration the heuristic is used. Let x be the current solution and x' be the new found solution. The following scaling parameter for updating π_h^i is introduced

$$\bar{\pi}_h^i = \bar{\pi}_h^i + 5^{\min(\sigma \cdot \text{gap},1)} \tag{6.5.3}$$

where $\text{gap} = \frac{c(x')-c(x)}{c(x)}$ and $\sigma \in \mathbb{R}^+$ is a parameter which needs tuning.

Acceptance criteria: The accept criteria used in this paper is borrowed from Ropke and Pisinger (2006) and is based on *Simulated Annealing* (SA). A solution, x is always accepted if $c(x) > c(x^{best})$. However if $c(x) \leq c(x^{best})$, x is accepted with the probability

$$\exp\left(-\frac{c(x^{best}) - c(x)}{T}\right) \tag{6.5.4}$$

where the *temperature* T is updated by $T = d_{\text{SA}} \cdot T$. Having $d_{\text{SA}} \in]0,1[$ as the *cooling rate*. The initial temperature is selected using a *temperature control* parameter, $w_{\text{SA}} \in]0,1[$, such that the solution is accepted with probability of 0.5 if the solution is w_{SA} percent worse than the initial solution x_0. Using an acceptance strategy where worse solutions can be accepted with a small probability makes it easier to expand and change neighborhoods.

Stopping criteria: The selection of removal and insertion heuristics are repeated until one of the following stopping criteria is met: (1) the running time exceeds the maximum running time of 60 seconds; or (2) the number of iterations without any improvements in the objectives reaches 1,000.

6.5.2 Removal and Insertion Heuristics

In this section the removal and the insertion heuristics used for the ECSS are described. Let $m \in \mathbb{N}$ be the number of classes which should be removed from a solution x and let $\bar{\mathcal{C}} \subseteq \mathcal{C}$ be the set of unassigned classes.

Random Removal Heuristic

The simple removal heuristic removes m elective classes with students from the solution. The classes are selected at random. This heuristic tends not to give better solutions, but it helps diversify the search. Furthermore a random removal which only removes classes which aren't locked, i.e. it does not remove classes which are a continuation from previous years.

Shaw Removal Heuristic

The general idea of Shaw removal heuristic is to remove parts of the solution which are somewhat related, as it is expected that they then are reasonably easy to reshuffle, and then creating a new, perhaps better solution (Shaw (1997),Ropke and Pisinger (2006)). Let the *relatedness measure* between meeting i and j be defined by $M(i;j) \in [0;1]$, where a high level corresponds to much

relatedness between i and j. Algorithm 5 presents the pseudo code for the Shaw Removal heuristic for the ECSS.

Algorithm 5: Shaw removal

Input: A feasible solution $x_{s,c,t}$, $m \in \mathbb{N}$, $p_{\text{shaw}} \in \mathbb{R}^+$

class: c = a randomly selected class with students from $x_{s,c,t}$
set of classes : $D = \{c\}$
while $|D| < m$ **do**
    $\hat{c}$ = randomly selected class from D
    L = all classes from $x_{s,c,t}$ not in D, sorted by similarity to $\hat{c}$
    choose a random number $b^{p_{\text{shaw}}} \in [0, 1[$
    l = element number $b^{p_{\text{shaw}}} \cdot |L|$
    $D = D \cup L[l]$
remove the classes with students in D from $x_{s,c,t}$

The Shaw removal heuristic of this paper is based on the number of students which have requested both course e of c and e' of c'. Let $\mathcal{S}_e$ indicates the students which have requested course e. The relatedness measure is then the percentage of students which have requested both courses of the two classes.

$$M(c, c') = \frac{|\mathcal{S}_e \cap \mathcal{S}_{e'}|}{\min(|\mathcal{S}_e|, |\mathcal{S}_{e'}|)} \qquad \text{where } D_{c,e} = D_{c',e'} = 1 \tag{6.5.6}$$

A high value of M means that the courses are much related.

Two Shaw heuristics for the ECSS is implemented. One sorted with increasing similarity, (i.e. removing those most related), and one with decreasing similarity (i.e. of those related, remove those less related).

Basic Greedy Insertion Heuristic

A basic greedy algorithm is implemented as one of the insertion heuristic for the ALNS. It simply assigns a class with students to a time slot in order for contribution to the objective. The process is repeated until no more classes with an improvement of the solution can be assigned.

The initial solution for the ALNS is constructed by means of a Basic Greedy Algorithm.

Regret-k Insertion Heuristic

The regret heuristic is a greedy algorithm with a look-ahead function incorporated, i.e. it tries to improve the myopic behavior of the greedy heuristic. As the name indicates, the heuristic aims at inserting the elective class which will be regretted most if not inserted at the given iteration. For each of the unassigned elective classes $\bar{c} \in \bar{\mathcal{C}}$, the regret-2 heuristic calculates a regret value equal to the difference in profit between two solutions in which $\bar{c}$ is assigned to its best time slot and its second best time slot. The unassigned class with the highest regret value, is the class which will be regretted most if not inserted in its best time slot, hence this is inserted. Let $o_{\bar{c}}^k$ denote the regret value by inserting class c into the k^{th} best position. The regret value of $\bar{c}$, $r_{\bar{c}}$, is given by

$$r_{\bar{c}} = \sum_{h_2}^{k} (o_{\bar{c}}^1 - o_{\bar{c}}^k) \tag{6.5.7}$$

In each iteration the heuristic chooses to insert class $\bar{c}$ according to $\max_{\bar{c} \in \bar{\mathcal{C}}} \{r_{\bar{c}}\}$ For the ECSS, it has been chosen to use at Regret-2, -3 and -4 as insertion methods.

Coupled Removal and Insertion Heuristic

In each iteration the ALNS heuristic chooses a removal and an insertion heuristic based on how well the pair has been performing previously. However some of the removal and the insertion

heuristics might not be a good matching due to the structure of the given heuristics. By pairing some of the heuristics, we simple declare which given insertion heuristic a removal heuristic should be paired with. For the ECSS we have created these coupled pairs, all only concerning students, i.e. only removing and inserting students, not classes. The other previous mentioned heuristics is concerning assigning/unassigning classes with students.

Three coupled pairs have been created. Let $\hat{E}$ be the set of courses where more than one class is created for the given course.

- Remove all students from classes in $\hat{E}$ and insert them using a basic greedy algorithm.
- Remove all students from classes in $\hat{E}$ and insert them greedily based on cohorts.
- Remove all students from classes in $\hat{E}$ located in the same given time slot $\hat{t}$ and insert them greedily.

6.5.3 Using Mathematical Programming within ALNS

The performance of the ALNS algorithm is evaluated by comparing it with solutions found using a state-of-the-art MIP solver (see Section 6.8). It can also be an advantage to include some mathematical programming (MP) techniques in the ALNS algorithm using a MIP solver. Heuristics which embedded MP methods are known as matheuristics.

In this paper we will try to embed mathematical programming techniques within the ALNS by introducing some new repair methods. We have only focused on the students as we did for the coupled constraints. I.e. the method removes all assigned students or all students assigned to classes in $\hat{E}$. All the classes are fixed. If some students are not removed, they are locked to the respective classes as well. Then by using a MIP solver we try to optimize the problem.

The performance of the ALNS with the MP techniques is evaluated in Section 6.8.2.

6.6 Defining Weights

The objectives of the problem are weighted in respect to each other, and the selection of the weights in the implementation in Lectio and for this article has been greatly assisted by MaCom A/S.

The profit of assigning a student to an elective course depends on the educational level of the course. Let $\alpha_{c,s} \in \mathbb{N}$ denote the profit of assigning student s to class c. Then $\alpha_{c,s}$ is given by

$$\alpha_{c,s} = \begin{cases} 95 & \text{If the course level of } c \text{ is at a } \textit{basic} \text{ level} \\ 100 & \text{If the course level of } c \text{ is at a } \textit{intermediate} \text{ level} \\ 105 & \text{If the course level of } c \text{ is at a } \textit{advanced} \text{ level} \\ 150 & \text{If } A_{c,s} = 1 \end{cases} \tag{6.6.1}$$

Notice that if the student is locked to the class the profit is quite high. This is to give preferential treatments to the classes which are a continuation from the previous year.

The cost of creating an elective class depends on the minimum number of classes which is necessary to fulfill all the requests for a given course. Let $MIN_e \in \mathbb{Z}^+$ be the minimum theoretical number of classes needed to fulfill all requests for course e. The cost of creating elective classes is then given by the following

$$\beta_c = \begin{cases} 150 & \rho(c) > Min_e \text{ where } D_{c,e} = 1 \\ 80 & otherwise \end{cases} \tag{6.6.2}$$

The cost of each represented cohorts in an elective class is given by

$$\gamma = 10 \tag{6.6.3}$$

For each student for which the two classes differ from each other is penalized by

$$\delta = 1 \tag{6.6.4}$$

6.7 Parameter Tuning

For tuning the free parameters of the heuristic, the *F-Race* algorithm has been chosen (Birattari, 2005). A race algorithm sequentially processes data instances using a set of all possible parameter configurations. After each iteration, the parameter configurations which are proven to be statistically inferior are eliminated from the set. In F-Race the *Friedman Two-way Analysis of Variance by Ranks* is used for determining whether any of the parameter configurations are statistically inferior. F-Race has previously been successfully used for parameter tuning for meta-heuristics (see. e.g. Chiarandini et al. (2006); Pellegrini and Birattari (2007); Kristiansen et al. (2013)). The drawbacks of F-Race are that all possible parameter configurations are considered, i.e. if many parameters with a wide range of values exist, the F-Race becomes inefficient and impractical. In Balaprakash et al. (2007) *Iterative F-Race* (I/F-Race) is introduced. I/F-Race uses a probabilistic model on the set of parameter configurations, such that only a subset of the parameter configurations is generated in each iteration.

In this paper a *manually* I/F-Race is used for tuning, i.e. after each iteration the new configurations are manually created based on the results from the previous iterations.

Table 6.1 lists the best found parameter configurations. Data instances from 50 different Danish high schools are used. For the SA based acceptance criteria two parameters are tuned. The temperature control, w_{SA}, and the decay parameter, d_{SA}. N_{it} defines the number of iterations between reset. For the ALNS scoring scheme, the tuned parameters are the reaction factor η and the scale parameter σ. ξ_{start} and ξ_{end} are the destroy percentage at the beginning and the end of the running time, respectively. Lastly p_{shaw} is the random indicator in the Shaw removal heuristic.

Table 6.1: Final values of tuned parameters, found by the F-Race algorithm with confidence level $\alpha = 0.05$.

Parameter	w_{SA}	d_{SA}	N_{it}	η	σ	ξ_{start}	ξ_{end}	p_{shaw}
Value	0.01	0.99	50	0.30	5000	0.30	0.0033	20

6.8 Performance

The purpose of this section is to evaluate the performance of the ALNS algorithm by comparing it with an upper bound found solving the IP model in the state-of-art MIP solver Gurobi 5.01. Both the ALNS and the Gurobi implementation was coded in C# 4.0 and all tests are performed using nUnit 2.6 on a machine with an Intel i7-930@2.8GHz CPU and 12GB of RAM under the Windows operating system. No parallelization has been implemented for improving the performance.

The ALNS algorithm for the ECSS presented in this article was launched for use in Lectio in mid-January 2012 and is as mentioned available for approximately 200 different high schools in Denmark. Up to this date over 500 data sets shared among the high schools, are available in the Lectio database.

6.8.1 Performance of ALNS compared with Gurobi

The ALNS algorithm is evaluated on 80 data sets. The data sets are selected randomly, and should cover all possible kinds of setups for the ECSS. The runtime is set to 60 seconds, which is the running time selected upon conversations between MaCom A/S and the users of Lectio. In order to reduce the eventual influence of stochastic behavior, 10 runs on each instance are performed. The Gurobi solver is run for 1 hour, as we want to have good upper bounds.

The percentage gap between the solution found using ALNS and the upper bound is calculated by $\frac{UB-\bar{x}_{ALNS}}{UB}$.

In Table 6.1 it is seen that the ALNS in average finds solutions less than 1% from optimum and some of the big instances the ALNS outperforms the solutions found using Gurobi. Moreover some of the instances are solved to optimality using ALNS. This is satisfying results.

Table 6.1: ALNS for the ECSS on 80 datasets compared with an upper bound using Gurobi 5.0.1. For each dataset is listed the number of students "$|\mathcal{S}|$", number of requests "$|\mathcal{R}|$" and number of courses "$|\mathcal{E}|$", which indicates the size of the given instance. For Gurobi is listed the final objective value, "x", the best bound "UB" and the reported gap between these. For the ALNS, the mean performance of the algorithm over 10 runs, "$\bar{x}$" and column "σ" is the standard deviation. Finally column "Gap(%)" is the percentage difference between ALNS and Gurobi.

				Gurobi 5.01			ALNS		
	$\|\mathcal{S}\|$	$\|\mathcal{R}\|$	$\|\mathcal{E}\|$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Aabenraa	20	20	3	1630.0	1630.0	0.0	1630.0	0.0	0.0
Aalborg	212	539	16	79010.0	79010.0	0.0	79010.0	0.0	0.0
Aarhus1	341	471	34	67745.0	67745.0	0.0	67745.0	0.0	0.0
Aarhus2	338	481	28	59134.0	59451.0	0.5	59099.6	4.8	0.6
Aars1	219	365	23	39315.0	39315.0	0.0	39315.0	0.0	0.0
Aars2	220	585	29	65615.0	65615.0	0.0	65615.0	0.0	0.0
Alssund	183	338	17	32827.0	33003.0	0.5	32824.6	3.6	0.5
Bagsvaerd1	49	75	10	10350.0	10350.0	0.0	10350.0	0.0	0.0
Bagsvaerd2	110	152	21	20210.0	20210.0	0.0	20210.0	0.0	0.0
Broenderslev1	312	515	22	49458.0	49943.0	1.0	49292.6	158.6	1.3
Broenderslev2	312	514	22	49357.0	49835.0	1.0	49089.9	149.1	1.5
CPHWEST1	249	426	32	48405.0	48498.0	0.2	48405.0	0.0	0.2
CPHWEST2	251	480	33	56041.0	56057.0	0.0	56041.0	0.0	0.0
DetFrie1	49	49	3	7110.0	7110.0	0.0	7110.0	0.0	0.0
DetFrie2	112	112	3	10839.0	10839.0	0.0	10825.0	0.0	0.1
Dronninglund1	299	522	27	75495.0	75495.0	0.0	75495.0	0.0	0.0
Dronninglund3	297	519	25	75380.0	75380.0	0.0	75380.0	0.0	0.0
Esbjerg	595	789	34	90050.0	90713.0	0.7	90006.6	27.7	0.8
EUCNORD	335	735	45	95080.0	95089.0	0.0	94993.4	78.3	0.1
Falkoner1	421	1080	49	131264.0	131728.0	0.4	130877.7	66.9	0.7
Falkoner3	649	1666	85	241015.0	241015.0	0.0	240641.0	788.5	0.2
Falkoner4	537	1376	57	177440.0	177674.0	0.1	177368.2	25.3	0.2
Falkoner5	431	617	42	64789.0	65210.0	0.7	64763.2	5.3	0.7
Falkoner6	297	456	34	41395.0	41828.0	1.1	41373.4	4.2	1.1
Falkoner7	742	1656	76	215578.0	216076.0	0.2	215446.0	53.0	0.3
Falkoner8	446	1266	56	160169.0	160362.0	0.1	160083.9	63.3	0.2
Falkoner10	335	520	34	50782.0	51428.0	1.3	50756.8	13.3	1.3
Fjerritslev	456	822	71	113706.0	113716.0	0.0	113701.2	2.5	0.0
Frederikssund1	294	475	25	53300.0	53300.0	0.0	53300.0	0.0	0.0
Frederikssund2	193	351	18	51130.0	51130.0	0.0	51130.0	0.0	0.0
Greve1	306	922	31	102351.0	103436.0	1.1	102347.8	89.8	1.1
Greve2	306	892	31	99500.0	100689.0	1.2	98980.8	246.5	1.7
Gribskov1	394	648	33	71895.0	72000.0	0.2	71325.6	129.7	1.0
Gribskov3	220	474	24	46632.0	46747.0	0.3	45968.0	296.2	1.7
GUAasiaat	71	82	12	11820.0	11820.0	0.0	11820.0	0.0	0.0
Haderslev1	447	1034	37	112065.0	113494.0	1.3	110729.1	1160.8	2.5
Haderslev2	470	1063	64	150480.0	150490.0	0.0	150480.0	0.0	0.0
Hasseris1	400	508	21	54019.0	54074.0	0.1	53846.0	92.2	0.4
Hasseris2	400	508	21	50859.0	51035.0	0.4	50384.8	92.8	1.3
HoejeTaastrup1	241	416	19	41010.0	41014.0	0.0	40909.4	80.9	0.3
HoejeTaastrup3	233	380	17	55330.0	55330.0	0.0	55330.0	0.0	0.0
Holstebro1	93	202	9	29420.0	29420.0	0.0	29420.0	0.0	0.0
Holstebro2	626	912	35	111064.0	111577.0	0.5	111021.0	6.7	0.5
Holstebro3	93	202	9	29420.0	29420.0	0.0	29420.0	0.0	0.0
Horsens	380	662	33	96660.0	96660.0	0.0	96660.0	0.0	0.0
Koebenhavns	289	816	31	100920.0	100930.0	0.0	100842.2	39.9	0.1
KoebenhavnsTek	166	169	7	24790.0	24790.0	0.0	24790.0	0.0	0.0
Koege	369	546	31	79360.0	79360.0	0.0	79360.0	0.0	0.0
Kongsholm1	383	760	46	109570.0	109570.0	0.0	107661.0	1655.6	1.8
Kongsholm2	365	974	40	126975.0	128862.0	1.5	126381.3	717.6	2.0
Langkaer	503	795	51	113540.0	113540.0	0.0	113540.0	0.0	0.0
Mariagerfjord1	365	521	24	75670.0	75670.0	0.0	75670.0	0.0	0.0
Mariagerfjord2	382	611	29	88170.0	88170.0	0.0	88170.0	0.0	0.0
Middelfart	390	1332	61	170243.0	170558.0	0.2	169663.3	232.0	0.5
Munkensdam	482	6456	231	349400.0	349400.0	0.0	349400.0	0.0	0.0
Noerresundby	563	1456	55	180916.0	181614.0	0.4	180807.5	50.1	0.5
NZahles	189	271	20	31955.0	31958.0	0.0	31955.0	0.0	0.0
Oeregaard1	239	489	13	71510.0	71510.0	0.0	71510.0	0.0	0.0
Oeregaard2	547	826	27	96376.0	97212.0	0.9	96425.0	11.6	0.8

Continued on next page

Table 6.1 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|\mathcal{S}\|$	$\|\mathcal{R}\|$	$\|\mathcal{E}\|$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Risskov1	539	784	38	91856.0	92529.0	0.7	91941.6	2.8	0.6
Risskov2	258	480	20	48993.0	49733.0	1.5	49038.8	3.8	1.4
Roedovre	350	868	34	100907.0	101414.0	0.5	100871.6	13.4	0.5
RoskildeKatedral	383	1145	40	119565.0	128359.0	7.4	126507.0	185.8	1.5
RoskildeTek1	358	529	21	77670.0	77670.0	0.0	77670.0	0.0	0.0
RoskildeTek2	358	688	30	91518.0	91520.0	0.0	91500.0	0.0	0.0
Rybners1	238	313	15	32454.0	32457.0	0.0	32336.6	130.0	0.4
Rybners2	352	443	11	40462.0	41245.0	1.9	40423.2	5.1	2.0
RybnersGym	192	384	20	55600.0	55600.0	0.0	55600.0	0.0	0.0
Rysensteen	285	570	19	56198.0	56429.0	0.4	55803.4	141.5	1.1
Skanderborg	245	439	18	41782.0	42246.0	1.1	41787.6	10.3	1.1
Slagelse1	974	1660	54	158025.0	180308.0	14.1	177314.5	129.9	1.7
Slagelse2	751	1345	45	127620.0	150895.0	18.2	148045.5	277.4	1.9
Slagelse3	1272	2221	57	131200.0	234645.0	78.9	229250.5	308.0	2.4
Slagelse6	1261	2289	113	329730.0	329730.0	0.0	329730.0	0.0	0.0
Slagelse7	329	508	23	63830.0	63836.0	0.0	63824.8	6.4	0.0
Struer	534	805	42	103161.0	103170.0	0.0	103140.2	5.4	0.0
Taarnby	298	760	29	110490.0	110490.0	0.0	110490.0	0.0	0.0
Varde2	230	677	30	98680.0	98680.0	0.0	98680.0	0.0	0.0
Vejlefjord	100	226	35	29985.0	29985.0	0.0	29985.0	0.0	0.0
Viby	232	472	18	47438.0	47798.0	0.8	47451.0	1.7	0.7
Average	359.5	756.6	34.4	-	-	1.7	-	-	0.5
Max	1272.0	6456.0	231.0	-	-	79.1	-	-	2.3

6.8.2 Performance using ALNS with Mathematical Programming Techniques

The MP methods are implemented as a repair method and as a hill climber, and both are tested using a running time of 2 and 5 seconds. For the hill climber this means that the running time for the ALNS is shortened by the running time of the hill climber such that the total running time still is 60 seconds.

The performance of the ALNS with some MP solution methods embedded are shown in Table 6.2. For the MP methods Gurobi 5.0.1 is used. For each solution method is listed the average gap and the max gap between the found solution and the best lower bound, taken over 80 datasets.

Table 6.2: Average performance using Gurobi and ALNS with different MP methods incorporated. The average is taken over 80 different dataset. The second column is the performance using Gurobi 5.01 and the third column is ALNS without any MP methods. Column 4 and 5 are average performance of the ALNS with a MP repair method. And Column 6 and 7 are the average performance using ALNS with a Hill Climber attached. Both tested with running time of 2 and 5 seconds

	Gurobi 5.01	ALNS	w. MP rep. (2 sec)	w. MP rep (5 sec)	w. MP HC (2 sec)	w. MP HC (5 sec)	w. MP rep & HC (2 sec)
Average	1.7	0.5	0.6	0.7	0.5	0.5	0.6
Max	79.1	2.3	2.7	4.1	2.7	2.2	3.4

As it is seen all the different ALNS algorithms outperform Gurobi on the average performance. This is due to the bad performance of Gurobi on the large instances. Furthermore, it is seen that the pure performs better than many of the ALNS with MP methods. This is mainly due to the running time and the scoring scheme. The running time has an important influence on the solution results. As the running time for the algorithm is 60 seconds, the running time of the MP repair methods cannot be too long. If it's more than 2-5 seconds it makes a significant decrease in the number of iterations we are able to perform. Yet, when having a low running time for the MP repair method it may results in poor performance of large instances and the solutions might not fulfill the acceptance criteria.

However, we can conclude that by embed some MP methods we might be able to improve the ALNS algorithm. For the ECSS using a MP hill-climber of 5 seconds at the end, improves the average performance a little.

6.9 Final Remarks and Outlook

In this paper the Elective Course Student Sectioning has been described in detail and formulated as an MIP model. ALNS has proven to be a successful method to establish solutions to the problem. The ALNS algorithm has been implemented in the Cloud-based software system Lectio and is hence available for more than 200 Danish high schools. For testing the performance of the ALNS algorithm, 80 real life instances from different high schools have been used. In average ALNS finds solutions within 1% of the optimum and for large instances the algorithm outperforms Gurobi, which is very satisfying results.

It was shown that for some of the instances it could be an advantage to embed some mathematical programming techniques in the ALNS. However more testing is needed with open source MIP solvers, as Gurobi is not a possibility as all the clients need a license.

Of future research within student sectioning at high schools it could be interesting to expand the model such that it contains the creation of the cohorts.

Acknowledgments

The authors thank Michael Bigom Herold from MaCom A/S for kindly helping determining the problem and setting the weights for the problem, and MaCom A/S for providing all the data.

Bibliography

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

M. W. Carter. A comprehensive course timetabling and student scheduling system at the university of waterloo. In E. Burke and W. Erben, editors, *Practice and Theory of Automated Timetabling III*, volume 2079 of *Lecture Notes in Computer Science*, pages 64–82. Springer Berlin Heidelberg, 2001. ISBN 978-3-540-42421-5.

M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9:403–432, 2006. ISSN 1094-6136.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775 – 1783, 2011. ISSN 0305-0548. doi: DOI:10.1016/j.cor.2011.02.007.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

L. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

L. F. Muller and S. Spoorendonk. A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem. Technical report, DTU Management Engineering, 2010.

P. Pellegrini and M. Birattari. Implementation effort and performance. pages 31–45. 2007.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

H. Rudova and K. Murray. University course timetabling with soft constraints. In *Practice And Theory of Automated Timetabling IV.*, pages 310–328, 2003.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

M. Sørensen and T. R. Stidsen. Integer programming and adaptive large neighborhood search for real-world instances of high school timetabling. *Annals of Operations Research*, PATAT 2012 SI:Submitted Jan 21. 2013, 2013.

M. Sørensen, S. Kristiansen, and T. R. Stidsen. International timetabling competition 2011: An adaptive large neighborhood search algorithm. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 489–492. SINTEF, 2012.

J. Steeg and M. Schröder. A hybrid approach to solve the periodic home health care problem. In J. Kalcsics and S. Nickel, editors, *Operations Research Proceedings 2007*, volume 2007 of *Operations Research Proceedings*, pages 297–302. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-77903-2.

D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967. doi: 10.1093/comjnl/10.1.85. URL `http://comjnl.oxfordjournals.org/content/10/1/85.abstract`.

Chapter 7

High School Student Sectioning at Danish High Schools

Simon Kristiansen*† Thomas R. Stidsen* Andrew Mason‡

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, thst@dtu.dk

† MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

‡Dept. of Engineering Science, University of Auckland
Auckland, New Zealand
a.mason@auckland.ac.nz

[1]

Abstract This article considers the *High School Student Sectioning* problem which is a recurrent planning problem at Danish high schools. Based on the students' choices of specializations and electives, the students need to be partitioned into cohorts and these cohorts assigned to time slots for the electives. The problem can be divided into two interconnected sub-problems: *cohort creation* and *elective timetabling*. An integer programming (IP) model for each sub-problem is presented and two solution approaches are suggested for solving the problem. In the first approach the two IP models are combined to produce a single optimal model which is solved using a modern IP solver. Tests on 25 real life problem instances show that the poor solutions generated using this approach make it unsuitable for use in a production system. The second approach solves the two sub-problems sequentially to produce a high quality heuristic initial solution for the combined model. Tests show that providing the solver with our heuristic starting solution significantly improves the quality of the solutions found, resulting in a system that is suitable for use by Danish high schools.

7.1 Introduction

High School Student Sectioning (HSSS) is a planning problem faced each year by high schools in Denmark and is concerned with the partitioning of new students into cohorts and assigning them to elective classes. This problem is of interest to the Danish software company MaCom A/S who develop cloud-based high school administration software *Lectio* used by the majority of high

[1]Submitted for publication in *European Journal of Operational Research*

schools in Denmark. As part of developing a new decision support tool for the HSSS problem, MaCom A/S and the authors have developed a problem specification through discussions with client schools. The HSSS problem definition we present here is the result of these discussions, and meets the requirements of all the high schools as to the size and format of their specific problems.

The outline of this paper is to present the HSSS in detail in Section 7.2 and build an IP model for the problem in Section 7.3. The solution methods are defined in Section 7.4. In Section 7.5 experiments and results for the HSSS are given. The conclusion and final remarks are given in Section 7.6.

7.2 High School Student Sectioning

In Denmark, first year high school students have two means by which they can control the subjects they wish to study. The first decision each student makes is to choose a *specialization*. These specializations are divided into so-called *Arts* and *Science* specializations. Examples of Arts specializations include, "English-German", "English-Music" and "English-Science", while the Science specializations include "Math-Physics" and "Math-Chemistry". Each student chooses exactly one specialization, which then determines a set of mandatory subjects (and hence taught classes) that all the students in that specialization need to take. Ideally, those students choosing the same specialization would form a single cohort in that they would be timetabled as a single entity, and thus take classes together. However imbalances in the numbers of students choosing each specialization means this is typically not possible, and so the first problem we face is to take, as input, the students and their specialization choices, and to produce from this a partitioning of students into cohorts that, wherever possible, group together those students with the same specialization in one cohort. We formally define and formulate this cohort creation problem in Section 7.2.1.

The second decision a Danish student makes is to choose two electives. Each student must select one *Linguistic* elective, such as "French" or "Italian", and one *Artistic* elective such as "Drama" or "Media". There are a number of time slots (i.e. periods of time) in the week that are set aside for the teaching of these electives. The second problem we face is to determine, for each student cohort, the two elective time slots to use for the teaching of that cohort's Linguistic and Artistic electives. When two or more cohorts are assigned the same elective time slot, students from the cohorts can be combined to form larger more efficient classes for each elective. Thus, this problem seeks to assign cohorts to elective time slots to minimize the number of elective classes that need to be run over the week. We address this *elective timetabling* problem in Section 7.2.2.

Thus, we see that the HSSS problem can be viewed as two sub-problems, the first being to partition students into cohorts and the second being to create efficient elective timetables for these cohorts. These two sub-problems are interconnected in that the efficiency of the elective timetabling solution depends on the cohort partitioning chosen in the first sub-problem. We consider the two sub-problems in detail in the next sections.

It is important to note that the HSSS forms part of a larger timetabling process operated by Danish high schools. The standard practice at these schools is to solve the student sectioning problem before a full timetable is generated. Thus, we are not, in this work, concerned with the timetabling of the non-elective classes, and indeed we do not know the actual times that the elective time slots will occupy. These decisions will be made in a subsequent optimization process that solves a general *High School Timetabling Problem* (Sørensen and Stidsen, 2013). This two-stage process has been requested by the schools as their preferred approach for integrating the optimization models as decision support tools into their larger decision making processes.

7.2.1 Partitioning to Create Cohorts

The first sub-problem is the partitioning of students to form *cohorts*. We define a cohort as a group of students that will be timetabled as one entity for most of their taught classes. High schools do not mix students from the Arts and Science specializations, and so the problem of creating student cohorts comprises two independent problems, one for students with an Arts specialization

and one for students with a Science specialization. A school will typically specify the number of Arts cohorts and the number of Science cohorts that need to be formed, and the minimum and maximum number of students allowed in each cohort.

We seek a solution in which each cohort contains students from just one or a small number of specializations. To formalize this, we say that a cohort contains a specialization if it includes one or more students who have chosen that specialization. A cohort has a specialization count given by the number of specializations it contains. The objective of our first sub-problem is to partition the students to form cohorts that minimize the sum, over the cohorts, of the cohort specialization counts.

Table 7.1 shows a possible solution for the Arts specializations for a representative high school. This high school has specified that we need to form five Arts cohorts each containing no more than 30 students and no fewer than 15 students. Table 7.1 shows a solution for the Arts specializations with a specialization sum of 7 over the 5 cohorts.

Table 7.1: An example of creating cohorts for students with Arts specializations. The upper limit on cohort size is 30 students while the lower bound is 15. The number of students in each specialization is given in the column marked "#", and the number in each cohort is shown in the 'Cohort Size' row. The columns marked "Coh1", "Coh2", ..., "Coh5" give the numbers of students assigned to the 5 different cohorts. The 'Spec. Count' row shows the specialization count for each chort.

Specialization	#	Coh1	Coh2	Coh3	Coh4	Coh5
English-German	18		18			
English-Spanish	40	28	12			
English-Music	9			9		
English-Science	77			20	30	27
Cohort Size		28	30	29	30	27
Spec. Count		1	2	2	1	1

7.2.2 Elective Timetabling

As detailed above, as well as choosing either an Arts or Science specialization, each student chooses one Linguistic elective and one Artistic elective. Each cohort must be assigned a single time slot for their Linguistic elective and another single time slot for their Artistic elective. These time slots are chosen from a set of predetermined Linguistic elective time slots and a set of predetermined Artistic elective time slots, respectively. These assignments then determine the number of students taking each elective in each time slot, which in turn determines the number of classes that must be created for each elective in that time slot to meet specified maximum class sizes. Ideally, we would assign all Arts electives to a single time slot, and all Science electives to another single time slot. However, the number of classes that can be run in a single time slot for an elective is limited, typically because of specialist classroom or teacher requirements. For example, if only two classrooms are suitable for the Music elective, it is not possible to have more than two Music classes in any one time slot. Thus, the elective timetabling sub-problem seeks an assignment of each cohort to a Linguistic elective time slot and to an Artistic elective time slot to minimize the total number of elective classes required while satisfying specified maximum class sizes and resource usage limits. Note that unlike the first sub-problem, which could be split into separate sub-problems for the Arts and Science specializations, this sub-problem must simultaneously consider all students across all Arts and Science specializations.

An example of Artistic elective choices for a representative set of cohorts is given in Table 7.2.

With an upper limit on elective class size of 30 a theoretical minimum of 9 elective classes are needed to fulfill the elective choices in Table 7.2. However this problem has a resourcing constraint specifying that no more than two Media classes can be run simultaneously, and thus this theoretical elective class count cannot be achieved. Table 7.3 shows a solution to the elective timetabling problem in Table 7.2 in which we have satisfied the Media resourcing constraint by creating 10 elective classes in total.

Table 7.2: An example of Artistic elective choices with 4 different Artistic electives (Painting, Music, Media, and Drama), 9 cohorts and a total of 224 students. The upper limit for class sizes is 30. The numbers in brackets denote the theoretical minimum number of classes needed for each elective for the number of students in that elective.

Cohort	Painting	Music	Media	Drama
Cohort 1	8	9	10	-
Cohort 2	7	3	11	9
Cohort 3	4	3	13	-
Cohort 4	11	3	6	10
Cohort 5	-	-	27	-
Cohort 6	6	1	14	-
Cohort 7	3	5	9	3
Cohort 8	5	6	14	-
Cohort 9	7	2	14	1
Total	51(2)	32(2)	118(4)	23(1)

Table 7.3: An example of a possible solution to the example given in Table 7.2 using three elective time slots (denoted #1, #2 and #3). The upper limit for class sizes is 30. The numbers in brackets are the number of classes created for the given elective in the given time slot. The last row shows the total number of students, and the total number of classes, for each elective.

Cohort	Painting	Music	Media	Drama
		Time slot #1		
Cohort 2	7	3	11	9
Cohort 4	11	3	6	10
Cohort 7	3	5	9	3
Cohort 9	7	2	14	1
#1 Total	28 (1)	13 (1)	40 (2)	23 (1)
		Time slot #2		
Cohort 1	8	9	10	-
Cohort 3	4	3	13	-
Cohort 6	6	1	14	-
Cohort 8	5	6	14	-
#2 Total	23 (1)	19 (1)	51 (2)	-
		Time slot #3		
Cohort 5	-	-	27	-
#3 Total	-	-	27 (1)	-
Total	51 (2)	32(2)	118 (5)	23(1)

7.2.3 Related Work

There has been an increase in scientific research within Educational Timetabling and High School Timetabling during the last decade, perhaps as a result of three international timetabling competitions: *University Timetabling* in 2003 (Kostuch, 2004), *University Timetabling and Examination Timetabling* in 2007 (Gaspero et al., 2007) and *High School Timetabling* 2011 (Post et al., 2012). However, the main focus of these competitions has been general timetabling problems and not student sectioning problems.

Those articles in the literature that address student sectioning mostly consider student sectioning at universities. In these cases the university timetable is generated first, and once a timetable has been developed, the object is to assign students to specific classes in order to minimize the number of conflicts. Many of the articles suggest heuristic procedures that iterate between the timetabling and the sectioning problem, see e.g. Carter and Laporte (1998) and Amintoosi and

Haddadnia (2005). However in a few articles a more integrated process is proposed where the student sectioning and the timetabling are solved as one problem. In Sönmez and Ünver (2010) they make use of course bidding systems. The students make requests (bids) for courses they wish to take. Based on these bids, the courses are allocated to classes and rooms in a way that reflects the number of bids for each course. Another approach is detailed in Müller and Murray (2010) where the timetabling and the sectioning problems are combined and solved using an Iterative Forward Search algorithm. Both of these papers focus on the timetabling problem and use the student sectioning part as guidance to the class and room allocation. Neither of these approaches considers the full student sectioning problem, and thus these are heuristics that cannot guarantee optimal solutions and give solutions of unknown quality.

The problem most closely related to our High School Student Sectioning Problem is the *Elective Course Planning Problem* (Kristiansen and Stidsen, 2013). Both our and their problems are yearly recurrent student sectioning problems at Danish high schools. In Kristiansen and Stidsen (2013) the problem is to assign 2nd and 3rd year students to electives given their requests, while our HSSS problem is concerned with the 1st year students and the partitioning of these students into cohorts.

7.3 Integer Programming Model

In this section, we develop integer programming models for HSSS. Consider a set of students, where each student has chosen one specialization from the set of specializations, $\mathcal{S}$, indexed by $s \in \mathcal{S}$, and two electives (one Linguistic and one Artistic elective) from the set of electives, $\mathcal{E}$. To remove some of the symmetry in the problem we introduce the set of student groups, $g \in \mathcal{G}$, where a student group g is a set of students who have requested the same specialization and the same two electives. In the first sub-problem of creating cohorts, we need to determine the number of students from each group $g \in \mathcal{G}$ to assign to each cohort $c \in \mathcal{C}$, where $\mathcal{C}$ is a set containing a predetermined number of cohorts. For the second sub-problem, we have a set of time slots $\mathcal{T}$ which are partitioned into two subsets, $\mathcal{T}_{\text{Linguistic}} \in \mathcal{T}$ and $\mathcal{T}_{\text{Artistic}} \in \mathcal{T}$ giving the available time slots for the Linguistic and Artistic electives respectively. Each cohort $c \in \mathcal{C}$ must be assigned to one Linguistic elective time slot $t \in \mathcal{T}_{\text{Linguistic}}$, and one Artistic elective time slot $t' \in \mathcal{T}_{\text{Artistic}}$.

It is helpful to consider the full HSSS problem as a set of decisions that can be viewed as generating a network such as that shown in Figure 7.1. These decisions include whether or not to include an arc in the solution and, for some arcs, the flows of students that occur on the included arcs. For the first sub-problem we let the decision variable $x_{g,c} \in \{0, 1\}$ take value one if any students from student group g are assigned to cohort c, and zero otherwise. We also introduce the flow variable $\bar{x}_{g,c} \in \mathcal{N}$ as defining the number of students from student group g assigned to cohort c. In the second sub-problem the cohorts are assigned to time slots by letting variable $y_{c,t,d} \in \{0, 1\}$ take value 1 if cohort c is assigned to time slot t for elective type $d \in \mathcal{D}$, $\mathcal{D} = \{\text{Linguistic}, \text{Artistic}\}$, and zero otherwise. As we will see shortly, the cost of a solution can be fully determined using these decision variables.

As mentioned previously the two sub-problems of HSSS are interconnected, as the elective timetabling sub-problem depends on the solution from the cohort creation sub-problem. However, for ease of explanation, we next develop optimization models for these sub-problems in which we assume these sub-problems are independent. These are then used to present a single model for the full problem. Note that we will present a single cohort creation model that includes both the Arts and Science specializations. This is required to develop the model for the full problem. However, as noted earlier, each specialization can be treated independently if we are solving this cohort creation sub-problem independently of the elective timetabling sub-problem.

7.3.1 Creating Cohorts

The first part of HSSS is the partitioning of students into cohorts, where we have to determine the flow $\bar{x}_{g,c}$ from each student group to each cohort, and a value for each associated binary variable

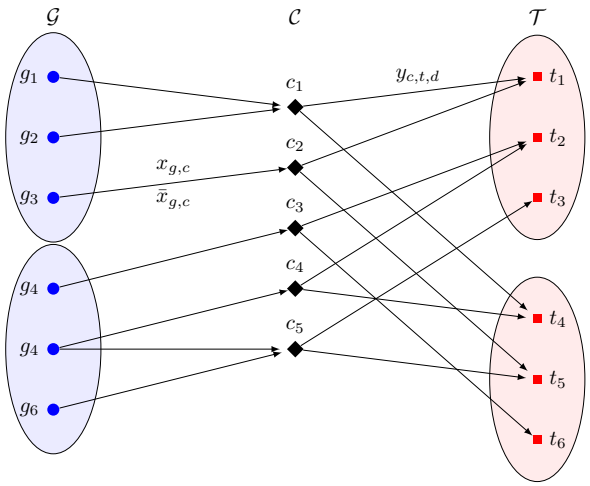


Figure 7.1: Illustration of the network flow representation of a solution of HSSS. The nodes on the left are the student groups, $\mathcal{G}$. The student groups are divided into two subsets, those with arts specializations $\{g_1, g_2, g_3\}$ and those with science specializations $\{g_4, g_5, g_6\}$. Student groups from these different subsets cannot be assigned to the same cohort. The second column lists the cohorts, $\mathcal{C} = \{c_1, c_2, \ldots, c_5\}$, for the problem. The third and last column lists the time slots, $\mathcal{T}$, which are divided into subsets $\mathcal{T}_{\text{Linguistic}} = \{t_1, t_2, t_3\}$ and $\mathcal{T}_{\text{Artistic}} = \{t_4, t_5, t_6\}$ giving the available Artistic and Linguistic elective time slots. Each cohort must be assigned exactly one time slot for each elective type. Where an arc is shown from a student group g to a cohort c, we have $x_{g,c} = 1$ and a non-zero flow in this arc given by $\bar{x}_{g,c}$. Similarly, an arc is shown from a cohort c to a time slot t for an elective type d for each $y_{c,t,d}$ =1.

$x_{g,c}$ that records if any non-zero flow occurs. To model each group's choices, we let the parameter $D_{g,s} \in \{0, 1\}$ be 1 if student group g has chosen specialization s, and zero otherwise, and let the $R_{g,e} \in \{0, 1\}$ be 1 if student group g has chosen elective e, and zero otherwise. Let the parameter $F_g \in \mathcal{N}$ be the size and hence the flow 'supply' of student group g, and let $A_{g,g'} \in \{0, 1\}$ be 1 if the student groups g and g' have chosen one Arts and one Science specialization, and thus are not allowed to be placed in the same cohort.

Let $U \in \mathcal{N}$ and $L \in \mathcal{N}$ be the lower and upper limit respectively on the cohort size as specified by the school. To ensure we can always find feasible solutions, even if these limits make the problem infeasible, we introduce a dummy cohort $c_\emptyset$ with no capacity limits, where the flow going into the dummy cohort, $\bar{x}_{g,c_\emptyset}$, gives the number of students which cannot be placed into a cohort. Note that this dummy cohort $c_\emptyset$ is not considered part of the set $\mathcal{C}$ of 'real' cohorts.

While our main goal is to minimize the specialization counts for the electives, the high schools also see it as advantageous if we can reduce the number of electives and different student groups represented in each cohort. To help model this in our objective, we introduce additional variables to further characterize the composition of each cohort. Specifically, we let $u_{s,c} \in \{0, 1\}$ and $z_{e,c} \in \{0, 1\}$ take value 1 if specialization s is represented in cohort c and if elective e is represented in cohort c, respectively, and zero otherwise.

The objective seeks to minimize a weighted combination of the numbers of student groups included in the cohorts, the number of unplaced students, the specialization counts for the cohorts, and the number of electives in the cohorts. The associated weights, α, β, δ and σ, are addressed more fully in Section 7.3.4.

The sub-problem of creating cohorts can now be written as follows:

Cohort Creation (7.3.2)

$$\min \quad \alpha \cdot \sum_{g\in\mathcal{G},c\in\mathcal{C}} x_{g,c} + \beta \cdot \sum_{g\in\mathcal{G}} \bar{x}_{g,c_\emptyset} + \delta \cdot \sum_{s\in\mathcal{S},c\in\mathcal{C}} u_{s,c} + \sigma \cdot \sum_{e\in\mathcal{E},c\in\mathcal{C}} z_{e,c} \tag{7.3.2a}$$

st

$$\sum_{c\in\mathcal{C}} \bar{x}_{g,c} + \bar{x}_{g,c_\emptyset} = F_g \quad \forall\, g \in \mathcal{G} \tag{7.3.2b}$$

$$\bar{x}_{g,c} \leq \min[F_g, U] \cdot x_{g,c} \quad \forall\, g \in \mathcal{G}, c \in \mathcal{C} \tag{7.3.2c}$$

$$\sum_{g\in\mathcal{G}} \bar{x}_{g,c} \leq U \quad \forall\, c \in \mathcal{C} \tag{7.3.2d}$$

$$\sum_{g\in\mathcal{G}} \bar{x}_{g,c} \geq L \quad \forall\, c \in \mathcal{C} \tag{7.3.2e}$$

$$x_{g,c} + x_{g',c} \leq 1 \quad \forall c \in \mathcal{C}, g, g' \in \mathcal{G}, g \neq g', A_{g,g'} = 1 \tag{7.3.2f}$$

$$D_{g,s} \cdot x_{g,c} \leq u_{s,c} \quad \forall\, g \in \mathcal{G}, c \in \mathcal{C}, s \in \mathcal{S} \tag{7.3.2g}$$

$$R_{g,e} \cdot x_{g,c} \leq z_{e,c} \quad \forall\, g \in \mathcal{G}, e \in \mathcal{E}, c \in \mathcal{C} \tag{7.3.2h}$$

$$x_{g,c} \in \{0,1\} \tag{7.3.2i}$$

$$\bar{x}_{g,c} \in \mathcal{N} \tag{7.3.2j}$$

$$u_{s,c} \in \{0,1\} \tag{7.3.2k}$$

$$z_{e,c} \in \{0,1\} \tag{7.3.2l}$$

Constraints (7.3.2b) ensure that all students in a student group are assigned to a real or dummy cohort. Constraints (7.3.2c) link the binary variables $x_{g,c}$ with the associated flows $\bar{x}_{g,c}$, ensuring that the number of students $\bar{x}_{g,c}$ from student group g assigned to cohort c is zero if $x_{g,c} = 0$, and otherwise $\bar{x}_{g,c}$ cannot be greater that the upper cohort size U or the size F_g of the student group. Constraints (7.3.2d) and (7.3.2e) make sure that the total flow into a cohort is between the lower and upper permitted cohort sizes, L and U. Constraints (7.3.2f) make sure that students with arts and science specializations are not placed in the same cohort. Finally constraints (7.3.2g) and (7.3.2h) set the values for the variables $u_{s,c}$ and $z_{e,c}$ and hence record if specialization s or elective e is represented in cohort c, respectively.

7.3.2 Elective Timetabling

The second part of the HSSS is the elective timetabling problem where we want to assign each cohort to one time slot for their Linguistic electives and one time slot for their Artistic electives. Thus, our inputs include the cohorts produced from the first sub-problem. As mentioned before, the decision variable $y_{c,t,d} \in \{0,1\}$ determines whether cohort c is assigned time slot $t \in \mathcal{T}$ for elective type $d \in \mathcal{D}$, $\mathcal{D} = \{\text{Linguistic}, \text{Artistic}\}$. (Note that electives of type $d \in \mathcal{D}$ can only be assigned to time slots $\mathcal{T}_d \subseteq \mathcal{T}$, and so many of these $y_{c,t,d}$ will be constrained to zero in our model.) We let variable $\bar{v}_{e,t} \in \mathcal{N}$ be the number of classes needed for elective e in time slot t and let variable $v_{e,t} \in \{0,1\}$ take value 1 if elective e is represented in time slot t, and 0 otherwise. The parameter $K_{e,t} \in \mathcal{N}$ denotes the maximum number of classes for elective e which can be created in time slot t, where each class in elective e can take up to $U_e \in \mathcal{N}$ students. Note that schools specify the exact number of elective time slots to be used for each elective type, and so we require each of these elective time slots to be used by at least one cohort.

The objective of this sub-problem is to minimize the number of elective classes required and to have only a few electives represented in each time slot, where these two objectives are weighted

using the cost parameters γ and ω (which are discussed further shortly). This gives the following model:

Elective Timetabling (7.3.3)

$$\min \quad \gamma \cdot \sum_{e\in\mathcal{E},t\in\mathcal{T}} \bar{v}_{e,t} + \omega \cdot \sum_{e\in\mathcal{E},t\in\mathcal{T}} v_{e,t} \tag{7.3.3a}$$

st

$$\sum_{t\in\mathcal{T}_d} y_{c,t,d} = 1 \quad \forall\, c\in\mathcal{C}, d\in\mathcal{D} \tag{7.3.3b}$$

$$\sum_{c\in\mathcal{C}} y_{c,t,d} \geq 1 \quad \forall\, t\in\mathcal{T}, d\in\mathcal{D} \tag{7.3.3c}$$

$$\sum_{g\in\mathcal{G}} R_{g,e}\cdot\bar{x}_{g,c} - M(1-y_{c,t,d}) \leq U_e\cdot\bar{v}_{e,t} \quad \forall\, c\in\mathcal{C}, e\in\mathcal{E}, t\in\mathcal{T}, d\in\mathcal{D} \tag{7.3.3d}$$

$$z_{e,c} + y_{c,t,d} - 1 \leq v_{e,t} \quad \forall\, e\in\mathcal{E}, c\in\mathcal{C}, d\in\mathcal{D} \tag{7.3.3e}$$

$$\bar{v}_{e,t} \leq K_{e,t} \quad \forall\, c\in\mathcal{C}, t\in\mathcal{T} \tag{7.3.3f}$$

$$y_{c,t,d} \in \{0,1\} \tag{7.3.3g}$$

$$\bar{v}_{e,t} \in \mathcal{N} \tag{7.3.3h}$$

$$v_{e,t} \in \{0,1\} \tag{7.3.3i}$$

Constraints (7.3.3b) make sure that every cohort is assigned exactly one time slot for each elective type. Constraints (7.3.3c) ensure that there is at least one cohort in each time slot, whereas Constraints (7.3.3d) set the value for each variable $\bar{v}_{e,t}$ given some appropriately chosen value for M. Constraints (7.3.3e) force elective e to be run in time t (i.e. $v_{e,t}$ =1) if any cohort with that elective is assigned to time slot t. Constraints (7.3.3f) ensure that the number of classes run simultaneously for an elective at a given time is within its bound.

If this sub-problem is solved independently, then $\bar{x}_{g,c}$ and $z_{e,c}$ are considered input parameters with values given by the solution of the first sub-problem. However, as we discusss next, a single model can also be formed by combing the two sub-models in which case $\bar{x}_{g,c}$ and $z_{e,c}$ are decision variables in the full model.

7.3.3 Full HSSS Model

Using the above definitions, we can create a 'full HSSS model' by combining the cohort creation sub-problem (7.3.2) and the elective timetabling sub-problem (7.3.3). This full model contains constraints (7.3.2b)-(7.3.2l) and (7.3.3b)-(7.3.3i) and seeks to minimize an expression given by the sum of the expressions in (7.3.2a) and (7.3.3a).

7.3.4 Defining Objective Weights

The HSSS is a multi-objective problem containing six different objectives. These objectives are weighted with respect to each other using a weighted sum. In Table 7.4 all the weights are listed with symbol, priority and cost.

It should be noted that feedback from a user will be used by Lectio to tune these weights to best reflect the particular needs of a school.

Table 7.4: Objective weights each with symbol, priority and value

Weight	Symbol	Priority	Value
Splitting student groups	α	Medium	10
Student not assigned	β	Very High	10000
Cost of created classes	γ	Medium	5
Specialization in cohort	δ	High	100
Course in cohort	σ	Low	2
Courses in time slot	ω	Low	1

7.4 Solution Methods

We want to test the models using two different solution methods. In the first 'standard' approach, we use an off-the-shelf IP solver to solve the full HSSS model. In the second 'sequential' approach, we solve the sub-problems in sequence to generate a good, but probably sub-optimal, heuristic initial solution. We then solve the full HSSS model using an off-the-shelf solver, but provide our heuristic solution as an initial incumbent solution. Our sequential heuristic is motivated by the empirical observation that the sub-problems are easy to solve when considered separately. Thus, our second solution method solves the HSSS problem using a sequential approach with three steps:

1. Solve the cohort creation sub-problem (7.3.2) for each specialization type, i.e. for Arts and for Science. As this sub-problem can be solved for each type of specialization independently, the two problems contain only a subset of the entire sub-problem.

2. Solve the elective timetabling sub-problem (7.3.3) using the fixed solution from the first step. As the students are already assigned to cohorts the second sub-problem cannot rearrange the students and hence the problem typically solves quickly.

3. Solve the full HSSS model using the solutions from the two previous steps as an initial incumbent solution.

By using a sequential approach we can solve the two IP models separately and fast and hence give a very good initial solution to the full combined model. It is therefore expected that the sequential approach will perform better than the standard approach for a given maximum run time. We note, however, that these methods are exact in that they will, if given sufficient time, both produce optimal solutions.

7.5 Experiments and Results

Implementations of the two approaches were created in in C# on a machine with an Intel i7 CPU, 3.07 GHz and with 12GB ram under Windows 8, 64 bit operating system. We used the state-of-art MIP solver Gurobi 5.5.0 with default parameter settings to solve the HSSS using the two different approaches.

MaCom A/S wish to use our models for decision support, and thus require good solutions quickly. They have set a maximum total running time of 60 seconds, and so, in the standard approach, we simply return the best solution found within this time. Our sequential process requires this running time to be divided between the steps. After some empirical experimentation, we found that good solutions could be generated by allocating 8 seconds to the first step (being 4 seconds for the sub-problem for each specialization), 2 seconds to the second step, and the remaining 50 seconds to the third step. The low running times allocated to the first two steps reflect the ease with which these problems can be solved when considered separately.

We tested the two approaches using 25 real life data instances of varying sizes sourced from Danish schools. These are listed in Table 7.5. To help benchmark our solutions, we generated 'best known' solutions for each instance using a modified form of our sequential approach in which,

during step 3, Gurobi was allowed to run for 4 hours. This gave us a best known objective value $\bar{z}$, a best known lower bound, $\bar{z}_{\text{LB}}$, and the gap between these:

$$\text{Gap}^{\bar{z}} = 100 \cdot \frac{\bar{z} - \bar{z}_{\text{LB}}}{\bar{z}}, \tag{7.5.1}$$

as shown in Table 7.5. These results show that these problems are hard for Gurobi to solve, producing bound gaps averaging 10.4% and up to 19.4% even after 4 hours. Of the 25 problems, only the two smallest problems (i.e. the two with the fewest students) were solved to optimality.

We next tested our standard approach in which the full HSSS is solved using Gurobi with a 1 minute maximum run time. Table 7.6 shows the results obtained using this standard approach. The resulting solution qualities are reported using the apparent percentage error, Err^{st}, measured between the solution objective z^{st} and the best known solution objective $\bar{z}$:

$$\text{Err}^{\text{st}} = 100 \cdot \frac{z^{\text{st}} - \bar{z}}{z^{\text{st}}}. \tag{7.5.2}$$

A user of the system will not know $\bar{z}$ or Err^{st}, but instead must assess the solution quality using a percentage bound gap Gap^{st} calculated using only data found during the run. Thus, we also report Gap^{st}, given by

$$\text{Gap}^{\text{st}} = 100 \cdot \frac{z^{\text{st}} - \text{LB}^{\text{st}}}{z^{\text{st}}}, \tag{7.5.3}$$

where LB^{st} is the lower bound reported by Gurobi at the end of the user's run. Before Gurobi solves the root node linear program, it uses heuristics to produce an initial integer solution giving a 'root node' objective function value $z^{\text{st}}_{\text{RN}}$. To assess the quality of Gurobi's heuristics, we report $z^{\text{st}}_{\text{RN}}$, the root node lower bound $\text{LB}^{\text{st}}_{\text{RN}}$, and the percentage gap at the root node given by

$$\text{Gap}^{\text{st}}_{\text{RN}} = 100 \cdot \frac{z^{\text{st}}_{\text{RN}} - \text{LB}^{\text{st}}_{\text{RN}}}{z^{\text{st}}_{\text{RN}}}. \tag{7.5.4}$$

Table 7.6 shows that solving HSSS using the standard approach gives poor performance producing bound gaps averaging 23.5% and up to 77% after 1 minute. Moreover, these solutions have apparent errors of 8.4% on average and as large as 73%. The quality of these solutions is too poor and varying for this approach to be viable as a production system.

Next we consider our sequential approach, results for which are given in Table 7.7. We have reported the same measures as used before, but note that the root node objective, now denoted $z^{\text{seq}}_{\text{RN}}$, is the objective function value of our heuristic solution (which Gurobi's heuristics never improved). Table 7.7 shows that our sequential approach produces solutions in 1 minute that are significantly better than a standard Gurobi implementation can generate in the same time. On average, our sequential approach reduces the average apparent error from 8.4% to 0.5%, and the average gap from 23.5% to 16.4%. The worst case apparent error drops from 73.2% to 2.4%, a significant improvement. Furthermore, our sequential method produces better initial solutions than are produced by Gurobi's pre-solve heuristics, reducing the root node gaps from 91.7% down to 24.8%. This approach has achieved our goal of producing good quality solutions within a 1 minute time frame, and thus is suitable for use by the Danish schools.

7.6 Conclusion

In this article we have shown how the High School Student Sectioning problem can be modeled as two separate, but interconnected, IP models. Two different approaches have been tested for solving the problem within the specified 1-minute maximum run time. The first standard approach is to solve and formulate a large IP model, while our second sequential approach decomposes the problem into two sub-problems that are solved sequentially to give the IP solver an initial heuristic solution. Testing on 25 real life instances has shown that the sequential approach significantly

Table 7.5: Specifications and best-known results for the 25 test instances. For each instance, this table shows the number of students —St—, the number of student groups $|\mathcal{G}|$, the number of specializations $|\mathcal{S}|$, the number of cohorts $|\mathcal{C}|$ and the number of Arts and Linguistic electives, denoted by $|\mathcal{E}_A|$ and $|\mathcal{E}_L|$ respectively. Furthermore, we have listed the best known objective $\bar{z}$, the best known lower bound $\bar{z}_{\text{LB}}$, and the percentage gap between these $\text{Gap}^{\bar{z}}$.

Instance	$\lvert St\rvert$	$\lvert\mathcal{G}\rvert$	$\lvert\mathcal{S}\rvert$	$\lvert\mathcal{C}\rvert$	$\lvert\mathcal{E}_A\rvert$	$\lvert\mathcal{E}_L\rvert$	$\bar{z}$	$\bar{z}_{\text{LB}}$	$\text{Gap}^{\bar{z}}$
Allerod	212	54	8	9	3	4	6869	5941	13.5
Aurehoj	147	38	11	6	3	4	4907	4631	5.6
Birkeroed	213	46	8	9	4	3	6089	5459	10.3
Borupgaard	279	42	14	11	3	4	6015	4880	18.9
Christianhavns	164	54	10	7	5	4	6697	6090	9.1
Falkoner	114	36	8	5	5	3	4594	4594	0.0
Fredericia	254	58	12	10	4	4	7528	6574	12.7
Gefion	237	66	12	10	4	5	8288	7232	12.7
GlHellerup	309	59	8	12	3	4	7850	6324	19.4
Haslev	205	75	12	8	5	4	9117	8107	11.1
Herlev	151	42	10	6	3	3	5292	4988	5.7
Hjoerring	261	67	12	10	3	5	8497	7247	14.7
HojeTaastrup	192	55	11	8	5	4	6950	6202	10.8
Kolding	176	55	8	7	4	3	6823	6285	7.9
Mulernes	153	31	10	6	4	4	4231	3975	6.1
Oerestad	273	70	11	10	4	4	8842	7695	13.0
Ordrup	210	48	10	9	3	4	6291	5582	11.3
Roskilde	180	58	13	7	4	4	7133	6542	8.3
Rybners	172	52	10	7	4	4	6436	5943	7.7
Rysensteen	227	29	11	10	2	6	4380	3555	18.8
Skanderborg	191	64	11	8	4	5	7947	7052	11.3
SktAnnae	65	21	7	4	4	3	3031	3031	0.0
Stenhus	257	71	13	10	4	6	8969	8082	9.9
Svendborg	273	62	13	10	3	4	7992	6977	12.7
VestFyen	179	47	8	7	4	3	5961	5496	7.8
Average	203.8	52.0	10.4	8.2	3.8	4.0			10.4

Table 7.6: Performance of the standard HSSS approach using Gurobi with a maximum 60 second run time. For each instance, the table shows the best known solution $\bar{z}$ from Table 7.5, the final objective value z^{st} found after 1 minute, the final lower bound LB^{st}, the gap between these Gap^{st}, and the percentage error Err^{st} between the best known and final solutions. The root node objective $z^{\text{st}}_{\text{RN}}$, root node lower bound $\text{LB}^{\text{st}}_{\text{RN}}$, and percentage gap $\text{Gap}^{\text{st}}_{\text{RN}}$ between these are also given.

		Standard approach						
Instance	$\bar{z}$	$z^{\text{st}}_{\text{RN}}$	$\text{LB}^{\text{st}}_{\text{RN}}$	$\text{Gap}^{st}_{\text{RN}}$	z^{st}	LB^{st}	Gap^{st}	Err^{st}
Allerod	6869	77574	5625	92.7	7101	5635	20.6	3.3
Aurehoj	4907	56585	4040	92.9	4907	4182	14.8	0.0
Birkeroed	6089	75060	4830	93.6	6165	4919	20.2	1.2
Borupgaard	6015	122742	4545	96.3	6124	4600	24.9	1.8
Christianhavns	6697	59397	5660	90.5	6842	5672	17.1	2.1
Falkoner	4594	39999	3830	90.4	4594	4101	10.7	0.0
Fredericia	7528	113150	6090	94.6	7660	6100	20.4	1.7
Gefion	8288	95678	6880	92.8	17781	6880	61.3	53.4
GlHellerup	7850	135306	6125	95.5	10502	6127	41.7	25.3
Haslev	9117	79070	7820	90.1	9358	7821	16.4	2.6
Herlev	5292	59121	4420	92.5	5351	4550	15.0	1.1
Hjoerring	8497	108890	6975	93.6	9367	6977	25.5	9.3
HojeTaastrup	6950	67909	5800	91.5	7335	5821	20.6	5.2
Kolding	6823	53208	5730	89.2	6922	5759	16.8	1.4
Mulernes	4231	65141	3345	94.9	4256	3459	18.7	0.6
Oerestad	8842	128133	7280	94.3	9237	7283	21.2	4.3
Ordrup	6291	78054	5065	93.5	6544	5091	22.2	3.9
Roskilde	7133	65010	6100	90.6	7423	6120	17.6	3.9
Rybners	6436	65357	5475	91.6	6471	5543	14.3	0.5
Rysensteen	4380	84111	3160	96.2	4407	3242	26.4	0.6
Skanderborg	7947	73788	6700	90.9	8017	6705	16.4	0.9
SktAnnae	3031	6379	2265	64.5	3031	3031	0.0	0.0
Stenhus	8969	114138	7444	93.5	33411	7561	77.4	73.2
Svendborg	7992	122548	6505	94.7	8996	6505	27.7	11.2
VestFyen	5961	58485	4925	91.6	6092	4952	18.7	2.2
Average				91.7			23.5	8.4

outperforms the standard approach on a range of measures, and is producing solutions with an average 16.4% optimality gap. Furthermore, our sequential approach gives solutions in 1 minute that are only 0.5% worse on average than the best solutions found within 4 hours, suggesting that

Table 7.7: Performance on the sequential HSSS approach using Gurobi. All the columns are defined in an analogous way to those in Table 7.6.

		Sequential approach						
Instance	$\bar{z}$	z_{RN}^{seq}	LB_{RN}^{seq}	Gap_{RN}^{seq}	z^{seq}	LB^{seq}	Gap^{seq}	Err^{seq}
Allerod	6869	6869	5625	18.1	6869	5638	17.9	0.0
Aurehoj	4907	4931	4040	18.1	4931	4116	16.5	0.5
Birkeroed	6089	6089	4830	20.7	6089	5162	15.2	0.0
Borupgaard	6015	122742	4545	96.3	6015	4617	23.2	0.0
Christianhavns	6697	6719	5660	15.8	6719	5711	15.0	0.3
Falkoner	4594	4846	3830	21.0	4594	4077	11.3	0.0
Fredericia	7528	7543	6090	19.3	7543	6312	16.3	0.2
Gefion	8288	8338	6880	17.5	8338	6906	17.2	0.6
GlHellerup	7850	7850	6125	22.0	7850	6129	21.9	0.0
Haslev	9117	9202	7820	15.0	9202	7845	14.7	0.9
Herlev	5292	5445	4420	18.8	5361	4604	14.1	1.3
Hjoerring	8497	8499	6975	17.9	8499	7006	17.6	0.0
HojeTaastrup	6950	7024	5800	17.4	7024	5874	16.4	1.1
Kolding	6823	6825	5730	16.0	6825	5774	15.4	0.0
Mulernes	4231	4260	3345	21.5	4260	3519	17.4	0.7
Oerestad	8842	8842	7280	17.7	8842	7308	17.3	0.0
Ordrup	6291	6291	5065	19.5	6291	5161	18.0	0.0
Roskilde	7133	7257	6100	15.9	7257	6167	15.0	1.7
Rybners	6436	6563	5475	16.6	6563	5525	15.8	1.9
Rysensteen	4380	84111	3160	96.2	4380	3296	24.7	0.0
Skanderborg	7947	8004	6700	16.3	8004	6707	16.2	0.7
SktAnnae	3031	3105	2265	27.1	3031	3031	0.0	0.0
Stenhus	8969	8969	7444	17.0	8969	7561	15.7	0.0
Svendborg	7992	8008	6505	18.8	8008	6546	18.3	0.2
VestFyen	5961	6107	4925	19.4	6107	4977	18.5	2.4
Average				24.8			16.4	0.5

the solution quality should meet the needs of the client schools. So in conclusion there is a huge advantage to solving the HSSS problem using our sequential approach.

The new sequential approach developed in this article is currently implemented as a beta system for Lectio. Our work has shown that we can model and solve this problem to deliver acceptably good solutions within the specified 1-minute run time, and thus this optimization approach is now ready for implementation as a new decision support tool within the Lectio system.

7.7 Acknowledgments

We would like to give our warm gratitude to MaCom A/S for their help in defining the High School Student Sectioning problem and for providing us with data for testing. This research has been partially supported by the European Union Seventh Framework Program (FP7-PEOPLE-2009-IRSES) under grant agreement number 246647 and by the New Zealand Government as part of the OptALI project.

Bibliography

M. Amintoosi and J. Haddadnia. Feature selection in a fuzzy student sectioning algorithm. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 147–160. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

L. D. Gaspero, A. Schaerf, and B. McCollum. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). Technical report, School of Electronics, Electrical Engineering and Computer Science, Queen's University SARC Building, Belfast, United Kingdom, 2007.

P. Kostuch. Timetabling competition - sa-based heuristic. In *PATAT 2004: Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, 2004.

S. Kristiansen and T. R. Stidsen. Elective course student sectioning at danish high schools. *Annals of Operations Research*, PATAT 2012 SI:To appear, 2013.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

G. Post, L. D. Gaspero, J. H. Kingston, B. McCollum, and A. Schaerf. The third international timetabling competition. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, Son, Norway, August 2012.

T. Sönmez and M. U. Ünver. Course bidding at business schools*. *International Economic Review*, 51(1):99–123, 2010. ISSN 1468-2354.

M. Sørensen and T. Stidsen. Comparing solution approaches for a complete model of high school timetabling. Technical Report 5.2013, DTU Management Engineering, Technical University of Denmark, March 2013.

Part IV

Meeting Planning Problems

Chapter 8

The Consultation Timetabling Problem at Danish High Schools

Simon Kristiansen*† Matias Sørensen*† Michael B. Herold† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
sikr@dtu.dk, msso@dtu.dk, thst@dtu.dk

†MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark
herold@macom.dk

[1]

Abstract In the different stages of the educational system, the demand for efficient planning is increasing. This article treats the $\mathcal{NP}$-hard *Consultation Timetabling Problem*, a recurrent planning problem for the high schools in Denmark, which has not been described in the literature before. Two versions of the problem are considered, the *Parental Consultation Timetabling Problem* (PCTP) and the *Supervisor Consultation Timetabling Problem* (SCTP). It is shown that both problems can be modeled using the same Integer Programming model. Solutions are found using the state-of-the-art MIP solver Gurobi and *Adaptive Large Neighborhood Search* (ALNS), and computational results are established using 300 real-life datasets. These tests show that the developed ALNS algorithm is significantly outperforming both Gurobi and a currently applied heuristic for the PCTP. For both the PCTP and the SCTP, it is shown that the ALNS algorithm in average provides results within 5% of optimum. The developed algorithm has been implemented in the commercial product Lectio, and is therefore available for approximately 95% of the Danish high schools.

8.1 Introduction

The *Consultation Timetabling Problem* (CTP) is a recurrent planning problem for the high schools in Denmark, which concerns the creation of a schedule for student-teacher meetings, given the students requests of teachers, subject to various soft constraints and resource constraints. The problem has not been described in the literature before, but it shares some properties with other problems within the educational sector, see Section 8.2.1. There exists several variants of the problem. In this paper we consider the two most important versions for the Danish high schools, namely the *Parental Consultation Timetabling Problem* (PCTP) and the *Supervisor Consultation Timetabling Problem* (SCTP).

[1]Published in *Journal of Heuristics, volume 19, 2013* (Kristiansen et al. (2013))

This paper is written in cooperation with the Danish company MaCom A/S. MaCom A/S is the developer of the cloud-based high school administration system Lectio, which handles all sorts of administrative tasks for the high schools, including a GUI and a heuristic-based solver for the PCTP. Through this cooperation we have access to real-life data for approximately 95% of all Danish high schools, which constitutes thousands of datasets. We will provide computational results for 300 of these datasets, which is a very big amount of real-life data compared to the majority of scheduling literature.

Our task of this paper is to give a detailed description of the CTP, and model it using Integer Programming. This model should support both the PCTP and the SCTP. To find solutions, both a state-of-the-art MIP solver and an *Adaptive Large Neighborhood Search* (ALNS) heuristic is attempted. These solution approaches are compared to the existing heuristic used in Lectio, and the best approach is made available for all users of Lectio.

8.2 Consultation Timetabling Problem

In the following we describe the CTP in details, starting with specifications of the two versions of the problem.

Parental Consultation Timetabling Problem: Once or twice a year the high schools invite the students and their parents to participate in meetings with the teachers of the student. The goal of these meetings is to allow the teachers to inform of the educational progress of the student, and possibly address relevant problems. Parental consultations usually take place in the evening of a normal work day, and each meeting generally has a duration between 5 and 15 minutes. The order of events for parental consultations is the following: The high school administration selects days where the meetings should take place, and for each day a feasible time-interval is chosen. Each student (usually in collaboration with his parents) makes prioritized requests of groups of teachers he would like to meet. Few of these teacher groups contain more than a single teacher, because the student is taught by only one teacher in most classes. Usually the high school also allows the students to request specific time intervals, within the overall time interval on each day, where the student will be available for meeting teachers. Given the student's choice of teachers and time intervals, it is then up to the high school administration to decide which teachers a student should meet, and when the meetings should take place.

Supervisor Consultation Timetabling Problem: In the last year of a high school education, the students are required to make a large study project (Danish: *Studieretnings Projekt*). Each student selects two subjects/courses as combined subject for his project, e.g. English and History. Each student is then assigned two teachers whom will be his supervisors for the project. The objective of the SCTP is to plan meetings between the students and their respective supervisors. The goal of these meetings is for the supervisors to provide the student with some useful hints for problem definition, literature research, etc. Typically supervisor consultations take place in the daytime, where both the student and the corresponding teachers are located at the high school.

From a timetabling point of view, these two types of consultations are almost identical. For both types, as many as possible of the meeting requests should be fulfilled, and both essentially contain the same constraints. Therefore we will in this paper model both types of consultations using the same Integer Programming model. In the remainder of this paper we refrain, as much as possible, from distinguishing between the two variants of the problem, and will by CTP denote the problem which is the superset of the PCTP and the SCTP.

In the following further details of the soft constraints of the CTP is given. These soft constraints define various scheduling preferences for the students and teachers.

A contiguous streak of meetings for a teacher or student are from now on denoted a *sequence*. A time slot is *void* for a teacher or student if the time slot is empty and no meetings are scheduled

in either earlier time slots or in later time slots. A *break* for a student or teacher is defined as a time slot which is not void, and which has no meetings assigned. Void time slots must be distinguished from breaks because they do not effect the density of a schedule. This is due to the fact that students and teachers are not obligated to stay at the school throughout the entire duration of the consultation. Given these definitions, we formulate the following soft constraints:

- It is attempted to achieve a solution where the positions of the granted meetings for a given individual are placed in a sequence. I.e. for both students and teachers the number of breaks should be minimized. This is to achieve a schedule with as little waiting time as possible. However, for the students it is possible for the high school administration to declare whether they need a break after each meeting. This is usually selected if there exists "traveling" time between the meeting rooms where the teachers are located. The CTP only takes consultation meetings into consideration when determining a sequence, and not other activities.

- When assigning a meeting to a time slot, the availability of the student and teacher must be taken into consideration. The high school administration decides whether this constraint should be defined as a hard- or a soft-constraint. It is common that in case of SCTP, this is defined as a soft-constraint as it is feasible for the students to leave classes to have meetings with their supervisors. In case of the PCTP, this constraint is usually treated as a hard-constraint, as a solution should respect activities such as meetings, study-trips, etc.

- It is undesirable for teachers to have too long sequences of meetings. Therefore a maximum on the length of sequences for teachers is given (treated as a soft-constraint). This is not required for the students, since they typically have a low number of requests.

- For a consultation which spans over several days it is desired that a student or a teacher only have meetings in one of the days, such that they are not obligated to attend both. This is especially critical for students, as they have a low number of requests.

- The high school administration prefers if the meetings are placed as close as possible to a specific time slot on each day. This is usually selected as the first timeslot on each day.

Figure 8.1 shows an example of a consultation schedule on one day. The schedule contains 1 void time slot, 2 breaks, and 7 consultations.

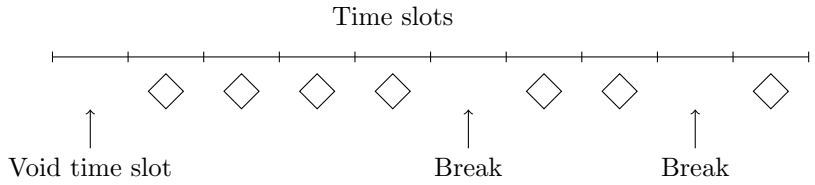


Figure 8.1: Example of a feasible consultation schedule.

8.2.1 Literature Review

The CTP has, to the best of our knowledge, not been described in the literature before. However there exist many types of related timetabling problems within the educational system, see Schaerf (1999); Burke and Petrovic (2002); McCollum (2006); Pillay (2010) for overviews of this field. Problems such as *Course Timetabling* and *Student Sectioning* have been looked in to, e.g. Tripathy (1984); Erben and Keppler (1996); Carter and Laporte (1998); Müller and Murray (2010). Related problems for Danish high schools include Kristiansen et al. (2011) and Kristiansen and Stidsen (2012) regarding the *Elective Course Planning Problem*, and Sørensen and Stidsen (2012) regarding

the timetabling problem. For all these problems it applies that they attempt to assign requests to time slots in a given schedule.

The requirement for compact schedules is well known in educational timetabling. In Santos et al. (2012) the Class-Teacher Timetabling Problem with Compactness Constraints is described. The compactness is defined in terms of teacher "holes", which is equivalent to our definition of breaks. The teacher holes are modeled with a linear IP, which entails the need for two auxiliary variables, and three additional constraints. This approach can be directly applied to the CTP. de Haan et al. (2007) specifies that the timetabling problem at Dutch high schools requires compact schedules for both classes and teachers. This is addressed using heuristic methods. For Greek high schools the situation is similar, see e.g. Birbas et al. (2009). Here the problem is handled using a MIP solver for a complicated IP.

8.3 Integer Programming Model

The following IP model for the CTP aims at maximizing the number of granted meeting requests and minimizing the violating of soft constraints, while respecting the hard constraints.

A consultation problem at a high school contains set of students S, a set of teachers T, a set of teacher groups G, an ordered set of time slots B and a set of of days D. $V_{b,d} \in \{0,1\}$ takes value 1 if time slot b is part of day d, and zero otherwise.

The decision whether a student s is given a meeting with teacher group g in time slot b is defined by the binary variable $x_{s,g,b} \in \{0,1\}$. The profit of meeting (s,g) in timeslot b is given by $\alpha_{s,g,b} \in \mathbb{R}^+$. The basic objective function is therefore

$$\max \sum_{s,g,b} \alpha_{s,g,b} x_{s,g,b} \tag{8.3.1}$$

8.3.1 Unavailabilities

In some situations it can be allowed to interrupt other activities at the high schools to satisfy a meeting request. Let $D_{t,b} \in \{0,1\}$ take value 1 if teacher t is not available (i.e. occupied by other activities) in time slot b, and zero otherwise. Let $E_{s,b} \in \{0,1\}$ be the completely analogous parameter for the students. If a consultation meeting is placed in a time slot where either a student or a teacher has some other activities, it is penalized by the following.

$$-\sum_{s,g,b} \left(\sum_{t} \delta_t \cdot D_{t,b} \cdot P_{g,t} + \delta_s \cdot E_{s,b} \right) x_{s,g,b} \tag{8.3.2}$$

If it is not allowed to interrupt activities this term is not added to the objective, and these constraints take the form of hard-constraints by forbidding meetings of teacher t in time slot b if $D_{t,b} = 1$, and likewise for student s in time slot b if $E_{s,b} = 1$.

It is of course not allowed to assign a student to a consultation meeting with a teacher group, if the student has not requested this teacher group. And is it not allowed for the student or teacher to have more than one meeting in each time slot. This imposed the following constraints. The parameter $P_{t,g} \in \{0,1\}$ takes value 1 if teacher t is in teacher group g, and zero otherwise. $R_{s,g} \in \{0,1\}$ takes value 1 if student s has requested teacher group g, and zero otherwise. $C_{s,b} \in \{0,1\}$ takes value 1 if student s has requested time slot b.

$$\sum_{b} x_{s,g,b} \leq R_{s,g} \qquad \forall\, s,g \tag{8.3.3}$$

$$\sum_{g} x_{s,g,b} \leq C_{s,b} \qquad \forall\, s,b \tag{8.3.4}$$

$$\sum_{s,g} P_{g,t} \cdot x_{s,g,b} \leq 1 \qquad \forall\, t,b \tag{8.3.5}$$

8.3.2 Undesirable breaks

One of the undesirable properties of the CTP is the breaks for both students and teachers. Let the variables $z_{s,d} \in \mathbb{N}$ and $w_{t,d} \in \mathbb{N}$ be the number of breaks in day d for a student and for a teacher, respectively. As we do not penalize void time slots as shown in Figure 8.1, we need to know when a individual have his first and last meeting. Let variables $f_{s,d}^{\text{first}}$ and $f_{s,d}^{\text{last}}$ denote the timeslot of the first and last meeting for student s, respectively. Let $h_{t,d}^{\text{first}}$ and $h_{t,d}^{\text{last}}$ be the analogous variables for teacher t. Let variable $n_{s,b,d} \in \{0,1\}$ take value 1 if student s is placed in time slot b on day d. The idle time slots for a student is then given by the following constraints

$$\sum_{g} V_{b,d} \cdot x_{s,g,b} = n_{s,b,d} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{8.3.6}$$

$$f_{s,d}^{\text{last}} - f_{s,d}^{\text{first}} + 1 - \sum_{b} n_{s,b,d} \cdot (1 + HS) + HS = z_{s,d} \qquad \forall\, s, d \tag{8.3.7}$$

$$|B|_d - (|B|_d - \text{ord}(b)) \cdot n_{s,b,d} \geq f_{s,d}^{\text{first}} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{8.3.8}$$

$$\text{ord}(b) \cdot n_{s,b,d} \leq f_{s,d}^{\text{last}} \qquad \forall\, s, b, d, V_{b,d} = 1 \tag{8.3.9}$$

The parameter $HS \in \{0,1\}$ indicates whether a student is required a break after each meeting, zero otherwise. This parameter is selectable for the user of Lectio. We want to penalize the cost such that it increases exponential on the number of breaks. The cost function is modeled as a piece-wise linear function by introducing a new variable $vs_{s,d,j} \in \{0,1\}$, where $j \in 1, ..., m$, which takes value 1 if student s has j breaks in day d. This imposes the following constraints.

$$\sum_{j} vs_{s,d,j} \cdot \text{ord}(j) = z_{s,d} \qquad \forall\, s, d \tag{8.3.10}$$

$$\sum_{j} vs_{s,d,j} = 1 \qquad \forall\, s, d \tag{8.3.11}$$

As for the teacher let variable $p_{t,b,d} \in \{0,1\}$ take value 1 if teacher t has a meeting in time slot b on day d. The cost for teacher breaks is also made as a piece-wise linear function, using the new variable $vt_{t,d,j} \in \{0,1\}$ The following constraints are imposed to denote the number of undesirable breaks for a given teacher,

$$\sum_{g,s} V_{b,d} \cdot P_{g,t} \cdot x_{s,g,b} = p_{t,b,d} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{8.3.12}$$

$$|B|_d - (|B|_d - \text{ord}(b)) \cdot p_{t,b,d} \geq h_{t,d}^{\text{first}} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{8.3.13}$$

$$\text{ord}(b) \cdot p_{t,b,d} \leq h_{t,d}^{\text{last}} \qquad \forall\, t, b, d, V_{b,d} = 1 \tag{8.3.14}$$

$$h_{t,d}^{\text{last}} - h_{t,d}^{\text{first}} + 1 - \sum_{b} p_{t,b,d} = w_{t,d} \qquad \forall\, t, d \tag{8.3.15}$$

$$\sum_{j} vt_{t,d,j} \cdot \text{ord}(j) = w_{t,d} \qquad \forall\, t, d \tag{8.3.16}$$

$$\sum_{j} vt_{t,d,j} = 1 \qquad \forall\, t, d \tag{8.3.17}$$

The contribution to the objective is as follows

$$-\sum_{s,d,j} \gamma_{s,j} \cdot vs_{s,d,j} - \sum_{t,d,j} \beta_{t,j} \cdot vt_{t,d,j} \tag{8.3.18}$$

8.3.3 Sequences

In connection with the undesirable breaks, the CTP also contains some needed breaks. For the students it is often necessary to give them a break between each consultation meeting due to traveling time between meeting rooms. This impose the following constraints.

$$\sum_{g} \left(x_{s,g,b} + x_{g,b_{+1},s}\right) \leq 1 \; \forall \, s, d, b \in B \backslash \{b_{|B_J|}\}, HS = 1, V_{b,d} = V_{b_{+1},d} = 1 \tag{8.3.19}$$

The teachers seldom change location during the consultation period, so travel time is not needed. However, as mentioned it is undesirable for the teachers to have a long sequence of meetings, as they need a break now and then. The maximum size of a sequence for a teacher is denoted $Q \in \mathbb{N}$. Let the variable $y_{t,b,d} \in \{0,1\}$ take value 1 if time slot b is the start of a sequence of length greater than Q on day d for teacher t. The following equation constraints this variable,

$$\sum_{s} \sum_{\substack{b'=b \\ V_{b',d}=1}}^{b+Q} p_{t,b',d} - y_{t,b,d} \leq Q \qquad \forall \, t, d, b \in B \backslash \{b_j | j > |B| - Q\}, V_{b,d} = 1 \tag{8.3.20}$$

The contribution to the objective is given by

$$-\sum_{t,b,d} \omega \cdot y_{t,b,d} \tag{8.3.21}$$

8.3.4 Day distribution

In case the consultation has more than one day it is preferred that each student and each teacher only has meetings on a single day. $u_{t,d} \in \{0,1\}$ and $u_{s,d} \in \{0,1\}$ denotes if teacher t or student s has a meeting on day d, respectively. $v_t \in \mathbb{N}$ and $v_s \in \mathbb{N}$ denotes the number of days where teacher t and student s have meetings, respectively. The number of days with meetings is punished in the objective by

$$-\sum_{s} \zeta_s \cdot v_s - \sum_{t} \zeta_t \cdot v_t \tag{8.3.22}$$

and is constrained by the following

$$\sum_{g} V_{b,d} \cdot x_{s,g,b} \leq u_{s,d} \qquad \forall \, s, b, d \tag{8.3.23}$$

$$\sum_{s,g} V_{b,d} \cdot P_{t,g} \cdot x_{s,g,b} \leq u_{t,d} \qquad \forall \, t, b, d \tag{8.3.24}$$

$$\sum_{d} u_{s,d} - 1 \leq v_s \qquad \forall \, s \tag{8.3.25}$$

$$\sum_{d} u_{t,d} - 1 \leq v_t \qquad \forall \, t \tag{8.3.26}$$

The entire model for CTP is given in (8.3.27).

8.3.5 IP model for CTP

Consultation Timetabling Problem IP (8.3.27)

$$\max \quad \sum_{s,g,b}\left(\alpha_{s,g,b} - \sum_{t}\delta_t \cdot D_{t,b} \cdot P_{g,t} + \delta_s \cdot E_{s,b}\right) \cdot x_{s,g,b} - \sum_{s,d,j}\gamma_{s,j} \cdot vs_{s,d,j} - \sum_{t,d,j}\beta_{t,j} \cdot vt_{t,d,j} \tag{8.3.27a}$$
$$- \sum_{t,b,d}\omega \cdot y_{t,b,d} - \sum_{s}\zeta_s \cdot v_s - \sum_{t}\zeta_t \cdot v_t$$

s.t.

$$\sum_{b} x_{s,g,b} \leq R_{s,g} \quad \forall\, s,g \tag{8.3.27b}$$
$$\sum_{g} x_{s,g,b} \leq C_{s,b} \quad \forall\, s,b \tag{8.3.27c}$$
$$\sum_{s,g} P_{g,t} \cdot x_{s,g,b} \leq 1 \quad \forall\, t,b \tag{8.3.27d}$$
$$\sum_{g}\left(x_{s,g,b} + x_{g,b_{+1},s}\right) \leq 1 \quad \forall\, s,d,b \in B\backslash\{b_{|B_J|}\}, HS = 1, V_{b,d} = V_{b_{+1},d} = 1 \tag{8.3.27e}$$
$$\sum_{s}\sum_{\substack{b'=b \\ V_{b',d}=1}}^{b+Q} p_{t,b',d} - y_{t,b,d} \leq Q \quad \forall\, t,d,b \in B\backslash\{b_j | j > |B| - Q\}, V_{b,d} = 1 \tag{8.3.27f}$$
$$\sum_{g} V_{b,d} \cdot x_{s,g,b} = n_{s,b,d} \quad \forall\, s,b,d, V_{b,d} = 1 \tag{8.3.27g}$$
$$|B|_d - (|B|_d - \mathrm{ord}(b)) \cdot n_{s,b,d} \geq f^{\mathrm{first}}_{s,d} \quad \forall\, s,b,d, V_{b,d} = 1 \tag{8.3.27h}$$
$$\mathrm{ord}(b) \cdot n_{s,b,d} \leq f^{\mathrm{last}}_{s,d} \quad \forall\, s,b,d, V_{b,d} = 1 \tag{8.3.27i}$$
$$f^{\mathrm{last}}_{s,d} - f^{\mathrm{first}}_{s,d} + 1 - \sum_{b} n_{s,b,d} \cdot (1 + HS) + HS = z_{s,d} \quad \forall\, s,d \tag{8.3.27j}$$
$$\sum_{j} vs_{s,d,j} \cdot \mathrm{ord}(j) = z_{s,d} \quad \forall\, s,d \tag{8.3.27k}$$
$$\sum_{j} vs_{s,d,j} = 1 \quad \forall\, s,d \tag{8.3.27l}$$
$$\sum_{g,s} V_{b,d} \cdot P_{g,t} \cdot x_{s,g,b} = p_{t,b,d} \quad \forall\, t,b,d, V_{b,d} = 1 \tag{8.3.27m}$$
$$|B|_d - (|B|_d - \mathrm{ord}(b)) \cdot p_{t,b,d} \geq h^{\mathrm{first}}_{t,d} \quad \forall\, t,b,d, V_{b,d} = 1 \tag{8.3.27n}$$
$$\mathrm{ord}(b) \cdot p_{t,b,d} \leq h^{\mathrm{last}}_{t,d} \quad \forall\, t,b,d, V_{b,d} = 1 \tag{8.3.27o}$$
$$h^{\mathrm{last}}_{t,d} - h^{\mathrm{first}}_{t,d} + 1 - \sum_{b} p_{t,b,d} = w_{t,d} \quad \forall\, t,d \tag{8.3.27p}$$
$$\sum_{j} vt_{t,d,j} \cdot \mathrm{ord}(j) = w_{t,d} \quad \forall\, t,d \tag{8.3.27q}$$
$$\sum_{j} vt_{t,d,j} = 1 \quad \forall\, t,d \tag{8.3.27r}$$
$$\sum_{g} V_{b,d} \cdot x_{s,g,b} \leq u_{s,d} \quad \forall\, s,b,d \tag{8.3.27s}$$
$$\sum_{s,g} V_{b,d} \cdot P_{t,g} \cdot x_{s,g,b} \leq u_{t,d} \quad \forall\, t,b,d \tag{8.3.27t}$$
$$\sum_{d} u_{s,d} - 1 \leq v_s \quad \forall\, s \tag{8.3.27u}$$
$$\sum_{d} u_{t,d} - 1 \leq v_t \quad \forall\, t \tag{8.3.27v}$$
$$x_{s,g,b} \in \{0,1\},\ y_{t,b,i} \in \{0,1\} \tag{8.3.27w}$$
$$w_{t,d} \in \mathbb{N},\ z_{s,d} \in \mathbb{N} \tag{8.3.27x}$$
$$vt_{t,d,j} \in \{0,1\},\ vs_{s,d,j} \in \{0,1\} \tag{8.3.27y}$$
$$f^{\mathrm{first}}_{s,d} \in \mathbb{N},\ f^{\mathrm{last}}_{s,d} \in \mathbb{N},\ h^{\mathrm{first}}_{t,d} \in \mathbb{N},\ h^{\mathrm{last}}_{t,d} \in \mathbb{N} \tag{8.3.27z}$$
$$p_{t,b,d} \in \{0,1\},\ n_{s,b,d} \in \{0,1\} \tag{8.3.27aa}$$
$$u_{s,d} \in \{0,1\},\ u_{t,d} \in \{0,1\} \tag{8.3.27ab}$$
$$v_s \in \mathbb{N},\ v_t \in \mathbb{N} \tag{8.3.27ac}$$

8.3.6 Complexity

In the following a proof of $\mathcal{NP}$-hardness is given by showing that a well known NP-hard problem, the Graph Coloring Problem (GCP), is polynomially reducible to CTP.

An arbitrary instance of GCP consists of a graph $G = (V, E)$ and a number of colors k. The decision-version of the GCP asks the following: Does graph G admit a proper vertex coloring with k colors, such that no adjacent vertices take the same color?

To answer this question we solve a CTP with parameters $\delta_t = \delta_s = \gamma_{s,j} = \beta_{t,j} = \omega = \zeta_s = \zeta_t = HS = 0, \alpha_{s,g,b} = 1, C_{s,b} = 1$. This makes all constraints redundant, except for (8.3.27b), (8.3.27c) and (8.3.27d). Further assuming every student has exactly one request, $\sum_g R_{s,g} = 1 \, \forall s$, makes constraint (8.3.27c) redundant.

For each vertex $v \in V$ in graph G, create a student s and a teachergroup g, and let the meeting request (s, g) represent vertex v. The set of vertices is hence represented by a setting of parameter $R_{s,g}$. If vertex $v_1 = (s_1, g_1)$ and vertex $v_2 = (s_2, g_2)$ are adjacent in graph G, create a teacher t and assign it to both g_1 and g_2, i.e. $P_{g_1,t} = P_{g_2,t} = 1$. I.e. every teacher will have exactly two meeting requests. Let the set of time slots B represent the set of colors (such that $|B| = k$).

Hence the GCP-instance is represented by the following CTP instance:

(8.3.28)

$$\begin{aligned}
\max \quad & \sum_{s,g,b} x_{s,g,b} && && && \text{(8.3.28a)} \\
\text{s.t.} \quad & \sum_{b} x_{s,g,b} && \leq && R_{s,g} \quad \forall\, s, g && \text{(8.3.28b)} \\
& \sum_{s,g} P_{g,t} \cdot x_{s,g,b} && \leq && 1 \quad \forall\, t, b && \text{(8.3.28c)} \\
& x_{s,g,b} \in \{0, 1\} && && && \text{(8.3.28d)}
\end{aligned}$$

Constraint (8.3.28b) specifies that each vertex (meeting request) can at most be assigned one color (time slot). Constraint (8.3.28c) specifies that no teacher can be assigned more than one meeting in each time slot, which specifies that no adjacent vertices can take the same color.

To answer the question whether G is k-colorable, solve the CTP instance (8.3.28) and check if all meeting requests are assigned a time slot, i.e. $\sum_b x_{s,g,b} = 1 \, \forall s, g, R_{s,g} = 1$. If so the answer is yes, otherwise the answer is no. Hence the Graph Coloring Problem is polynomially reducible to CTP, and CTP is therefore $\mathcal{NP}$-hard.

8.3.7 Defining Weights

In the following the weights of the model are selected due to the preferences of the Danish high schools. MaCom A/S has greatly assisted this process. Table 8.29 lists all the weights in the model and their priority.

From analysis of previous consultations in the Danish high schools, it is noticed that the students rarely request more than five teacher groups for consultations. And even though the students have the opportunity to request more than five, they seldom use this option. From this analysis it is chosen to stick the request with priority higher than five to the same weight. This gives the following function for the request weights $\alpha_{s,g,b}$:

$$\alpha_{s,g,b} = \begin{cases} \kappa_b + 12 - 2 \cdot (i - 1) & i \leq 5 \\ \kappa_b + 2 & i \geq 6 \end{cases} \tag{8.3.30}$$

where $i \in \mathbb{Z}^+$ is the priority of request (g, s). Furthermore there is a set-point for each day given by b_d^* for which it is desired that the schedule plan for the given day is centered around. Let κ_b

Table 8.29: Weight prioritizing

Weight	Symbol	Priority	Value dependency
(Max) Request fulfilling	$\alpha_{s,g,b}$	Very High	Priority of request (s, g) in time slot b
(Min) Teacher holes	$\beta_{t,j}$	High	Amount of requests of teacher t
(Min) Teacher sequence violation	ω	High	N/A
(Min) Teacher activities interruption	δ_t	Medium	N/A
(Min) Teacher multiple days	ζ_t	Medium	N/A
(Min) Student holes	$\gamma_{s,j}$	High	Amount of requests of student s
(Min) Student activities interruption	δ_s	Medium	N/A
(Min) Student multiple days	ζ_s	Medium	N/A

denote the penalty for assigning a request to time slot b, defined by

$$\kappa_b = -\frac{\sum_d V_{b,d} \cdot |b_d^* - b|}{|B|} \tag{8.3.31}$$

The cost of an undesirable break for a teacher, $\beta_{t,j}$, is defined as follows,

$$\beta_{t,j} = \begin{cases} 0 & j = 0 \\ j & j \geq 1 \wedge SCTP \\ j^{1+\frac{1.5}{\eta_t}} & j \geq 1 \wedge PCTP \end{cases} \tag{8.3.32}$$

where η_t is the number of requests for teacher groups where teacher t is a group member, $\eta_t = \sum_{s,g} P_{g,t} \cdot R_{s,g}$. I.e. $\beta_{t,j}$ depends on the number of requests for the given teacher t. The distribution of $\beta_{t,j}$ is chosen such that a teacher which few students have requested is given a high penalty for undesirable breaks. Likewise, a teacher with many requests has a low penalty for undesirable breaks. This is due to the fact that teachers with many requests will most likely have a more dense schedule, and are therefore not too picky about additional breaks. The reason why there is a difference between the weights for the different consultations types is due to the consultation interval. For the PCTP the consultation meetings are normally located in the evening, and hence we want to penalize the undesirable breaks. The SCTP is typically taken place in the daytime, i.e. the teachers are already at the high school, hence undesirable breaks are not that significant. The weight of an undesirable break for a student $\gamma_{s,j}$ is analogues,

$$\gamma_{s,j} = \begin{cases} 0 & j = 0 \\ j^{1+\frac{2}{\eta_s}} & j \geq 1 \end{cases} \tag{8.3.33}$$

where η_s is the number of requests of student s.
The cost for violating the length of a sequence for a teacher is given by ω.

$$\omega = 2 \tag{8.3.34}$$

In our model of the CTP the high school administration selects if interrupting other activities of the students or teachers is allowed. If this is not the case, the costs δ_t and δ_s are selected as infinitely high (implementation-wise the corresponding constraints are treated as hard-constraints). If interrupting other activities are not allowed, these costs are selected as a constant value,

$$\delta_t = \delta_s = \begin{cases} \infty & \text{Interrupting activities not allowed} \\ 4 & \text{Interrupting activities allowed} \wedge PCTP \\ 1 & \text{Interrupting activities allowed} \wedge SCTP \end{cases} \tag{8.3.35}$$

Like for the undesirable break cost $\beta_{t,j}$, we distinguish between the two consultations types. For the PCTP it is expensive to interrupt other activities since it is held in the evening and hence other activities are typically other types of meetings. The SCTP is held in the daytime, and it is allowed to 'lent' a student from a lecture for a small cost.

8.4 Adaptive Large Neighborhood Search

In this section a heuristic alternative to solving the IP-model (8.3.27) is described. The performance of these two methods are compared in Section 8.6.

As the local search algorithm we have chosen to use the *Large Neighborhood Search* (LNS) proposed by Shaw (1998). Most local search algorithms explicitly defines the neighborhood, but the neighborhood in LNS is defined implicitly by a *destroy* and a *repair* method. The neighborhood of a solution is then defined as the set of solutions that can be reached by first applying a destroy method and then a repair method. In this article we will use Adaptive Large Neighborhood Search (ALNS), in which the LNS is extended by multiple destroy and repair methods. ALNS was first described in Pisinger and Ropke (2005), and has since been used with success on various problems, especially variants of Vehicle Routing Problems (VRP), see e.g. Ropke and Pisinger (2006); Laporte et al. (2010); Azi et al. (2010); Lei et al. (2011); Ribeiro and Laporte (2012). A pseudo-code for the ALNS heuristic is shown in Algorithm 1.

Algorithm 1: Adaptive Large Neighborhood Search

Input: a feasible solution $x^s_{g,b}$
1 solution $x_{best} = x$; $\pi = (1, \ldots, 1)$
2 **repeat**
3 $x' = x$
4 select destroy and repair methods $d \in \Omega^-$ and $r \in \Omega^+$ using π
5 select $q \in \mathbb{N}$
6 remove q requests from x' using d
7 reinsert removed requests into x' using r
8 **if** $c(x') > c(x_{best})$ **then**
9 $x_{best} = x'$
10 **if** *accept(x', x)* **then**
11 $x = x'$
12 update π
13 **until** *stop-criterion met*
14 **return** x_{best}

The sets of destroy and repair methods are denoted Ω^- and Ω^+, respectively. The variable π, which stores the weight of all destroy and repair methods, is introduced in line 1. Initially all methods have the same weight. In line 1 the weight parameter π is used to select the destroy and repair methods. In line 1 an accept function evaluates if the new solution should become the new current solution. The accept function can be implemented in different ways. We have chosen to implement a Simulated Annealing-like acceptance criterion, which will be described later.

An ALNS framework has the advantage of using different neighborhoods, such that the algorithm hopefully explores a large part of the solution space. For more information regarding ALNS we recommend Ropke and Pisinger (2006) and Pisinger and Ropke (2010).

8.4.1 ALNS Scoring Scheme

A central part of the ALNS algorithm is the scoring scheme of destroy and repair methods. A scoring scheme can essentially be characterized by two central topics; 1) How to quantify the performance of each heuristic. 2) The reaction factor, i.e. how sensitive is the selection process to recent records of performance.

We adapt a scoring scheme based on the technical report of Muller and Spoorendonk (2010), where performance is tracked by the percentage-wise gap between the new found solution and the current solution. This scoring scheme has the advantage of having few parameters to tune, and using the gap between solutions seems as a intuitively good way of measuring heuristic performance. Below the scoring scheme is explained in details.

Runs of the algorithm is divided into segments $\{t_0, t_1, \ldots, t_n\}$ each consisting of N_{it} iterations. Let π_i^t be the weight of heuristic i in segment t. The probability of choosing heuristic i in segment t is $\frac{\pi_i^t}{\sum_j \pi_j^t}$. At the end of each segment t, the following update is performed for all heuristics,

$$\pi_i^{t+1} = \rho \frac{\bar{\pi}_i^t}{a_i^t} + (1-\rho)\pi_i^t \tag{8.4.2}$$

where a_i^t is the number of times heuristic i has been selected in segment t. $\bar{\pi}_i^t$ is the *observed* weight of heuristic i in segment t, which in each iteration is incremented depending on the quality of the new found solution. $\rho \in [0,1]$ is the reaction factor. A high reaction factor means that the weights of a segment will be very dependent upon the observed weights of the previous segment.

The observed weight $\bar{\pi}_i^t$ is updated in each iteration. Let x be the current solution, and x' the new found solution by applying neighborhood i. In the technical report Muller and Spoorendonk (2010) the following formula is used,

$$\text{gap} = \frac{c(x') - c(x)}{c(x)} \tag{8.4.3}$$

$$\bar{\pi}_i^t = \bar{\pi}_i^t + m^{\text{gap}} \tag{8.4.4}$$

where m is a constant. We will use a slightly changed version of this formula, since we have observed that the gap formulated by (8.4.3) most often yields values of magnitude $\pm 10^{-4}$, meaning that the observed weight $\bar{\pi}_i^t$ will rarely change value of significant magnitude. Therefore we introduce a scale parameter in the formula,

$$\bar{\pi}_i^t = \bar{\pi}_i^t + m^{\min(\sigma \cdot \text{gap}, 1)} \tag{8.4.5}$$

where $\sigma \in \mathbb{R}^+$ is a parameter that needs tuning. We fix $m = 5$ and rely on the parameter tuning to set a suitable value for σ. The min-operator in the exponent of m is necessary to ensure the weight stay within a reasonable interval, in case we hit an iteration where the scaled gap is big and positive.

8.4.2 Request Removal

The ALNS heuristic for the CTP makes use of two different removal heuristics, each searching a given removal neighborhood. The heuristics takes as input a given solution $x_{s,g,b}$ and an integer $q \in \mathbb{N}$. The output of the heuristics is the solution where q meetings have been removed. The value of q is selected as a random number which satisfies,

$$3 \leq q \leq \max\left(\xi \cdot \sum_{g,s} R_{s,g}, 5\right) \tag{8.4.6}$$

where $\xi \in]0,1]$ is the maximum percentage of requests to remove. In accordance with Muller (2009) we decay ξ over time, starting with a high value ξ_{start} and ending with a smaller ξ_{end}. Given the runtime of the algorithm, we divide it into 100 segments such that ξ is decreased by $\frac{\xi_{\text{start}} - \xi_{\text{end}}}{100}$ in each segment. This decay of ξ means that the size of the searched neighborhood is progressively reduced. This has the advantage of only performing small changes towards the end of the solution process, where we expect a good solution has been found.

Random removal

The simplest removal heuristic, which randomly removes q meetings from the solution. This simple heuristic obviously has the effect of diversifying the search.

Shaw removal

This removal heuristic was introduced in Shaw (1997, 1998) where it is used on the VRP. In this section the heuristic is modified to suit the CTP. The general idea of the heuristic is to remove meetings which are somehow related, since there is a good chance that such requests can swap positions and possibly improve the solution. In this paper two factors determine if meeting i is related to meeting j: Similarity between students, and similarity between teachers. S_i and T_i indicates the set of students and teachers of meeting i, respectively. Notice that a meeting always contains exactly one student, i.e. $|S_i| = 1$. Let the measure of relatedness between meeting i and j be defined by $\mathcal{M}(i,j) \in [0,1]$,

$$\mathcal{M}(i,j) = \frac{|T_i \cap T_j| + |S_i \cap S_j|}{\min(|T_i|, |T_j|) + 1} \tag{8.4.7}$$

I.e. relatedness is the percentage of individuals which is shared between the meetings, such that a high value of $\mathcal{M}$ means that the meetings are very related. This simple formulation of relatedness is done without any additional parameters. An alternative natural formulation would be to scale the student-relatedness and teacher-relatedness by two independent parameters. However we have chosen the shown formula due to its simplicity. An addition to the formula could be to introduce a term which determines time slot relatedness, although it should be noted that relatedness between slots is only directly relevant if there also exists some relatedness between students or teachers.

A pseudo code for Shaw removal is shown in Algorithm 8.

Algorithm 8: Shaw removal

Input: A feasible solution $x_{s,g,b}$, $q \in \mathbb{N}$, $p_{\text{shaw}} \in \mathbb{R}^+$
request: r = a randomly selected meeting from $x_{s,g,b}$
set of requests : $D = \{r\}$
while $|D| < q$ **do**
    r = randomly selected meeting from D
    L = all meetings from $x_{s,g,b}$ not in D, sorted by decreasing similarity to r
    choose a random number $y^{p_{\text{shaw}}} \in [0,1[$
    l = element number $y^{p_{\text{shaw}}} \cdot |L|$
    $D = D \cup L[l]$
remove the meetings in D from $x_{s,g,b}$

To avoid the situation where the same meetings are removed over and over, the algorithm is randomized. The level of randomness is controlled by the parameter $p_{\text{shaw}} \in \mathbb{R}^+, p_{\text{shaw}} \geq 1$. This means that p_{shaw} somehow defines how random the element is chosen, where $p_{\text{shaw}} = 1$ corresponds to completely random.

8.4.3 Repair Heuristic

The repair heuristics are given a set of consultation meetings and a set of not granted meeting-requests.

Basic greedy heuristic

A trivial algorithm for the CTP is a simple greedy algorithm which places one request at a time in order of contribution to the objective. In each iteration of the algorithm this process is repeated until no more requests which improves the solution can be inserted. Implementation wise the algorithm suffers from cost-dependencies, since the contribution of inserting each request possibly changes after each insertion. This is slightly optimized by only recalculating the cost of those requests which the last insertion can possibly effect. I.e. recalculate the cost of those requests which has the same student as the last insertion, or if the teacher group overlaps with the one of the last insertion. This repair heuristic is used to create an initial feasible solution for the CTP.

Regret heuristics

The regret heuristic improves the basic greedy by incorporating a kind of look-ahead information when selecting a request to insert. Informally speaking, the heuristic aims at inserting the request which we will regret most if not inserted immediately. The regret heuristic has been used by Potvin and Rousseau (1993) and Pisinger and Ropke (2005) for the *Vehicle Routing Problems with Time Windows*. Let c_r^k denote the change in the objective value by inserting request r into the k^{th} best position. E.g. c_r^2 denotes the change in the objective value by inserting request r in the second best position. A Regret-2 heuristic will in each iteration choose to insert the request r where the difference between best and second best position is largest, i.e.

$$r := \underset{r \in R_g^s, c_r^1 > 0}{\arg\max} \left(c_r^1 - c_r^2\right) \tag{8.4.9}$$

The request r is inserted at its best position, so we restrict the heuristic to only look at requests where the best position is actually feasible and yields a positive change in objective. This restriction is necessary since the objective of the CTP contains both a minimization and a maximization part, and we are not interested in inserting requests which have negative impact on the objective. The heuristic can be extended by looking at k positions for each request. The request to insert is then chosen according to

$$r := \underset{r \in R_g^s, c_r^1 > 0}{\arg\max} \sum_{h=2}^{k} \left(c_r^1 - c_r^h\right) \tag{8.4.10}$$

We will in this paper incorporate the regret heuristic for several choices of k. The basic greedy algorithm from the previous section is a Regret-1 heuristic due to the tie-breaking rules. For a Regret-1 heuristic the most profitable request is inserted in each iteration. Most papers distinguish between Regret-1 and other regret heuristic, however implementation wise they are not very different. Setting $k = |B|$ corresponds to the full Regret-k heuristic.

Even though the regret heuristic is designed for VRP, it seems well suited for the CTP due to its assignment character. It seems valuable to attempt to predict which request is the most critical to insert. By some basic tests, we have chosen to use Regret-2, Regret-3, and Regret-$|B|$ as insertion heuristics.

8.4.4 Algorithm Setup

According to Ropke and Pisinger (2006), using myopic repair heuristics, like those of this paper, one may apply noise to the objective function to obtain a more efficient algorithm. By applying noise, the repair heuristic will not always make the move that seems best locally. Ropke and Pisinger (2006) support this by strong computational results. However, preliminary tests show that, in our case, adding noise does not yield a more efficient algorithm. More precisely, noise was added such that it was controlled by a linear-scale parameter, and excessive tuning on this parameter yielded no convergence at all. I.e. this parameter had (close to) no impact on the algorithm efficiency. A similar result for the *Cumulative Capacitated Vehicle Routing Problem* is reported in Ribeiro and Laporte (2012).

In occurrence with Ropke and Pisinger (2006) we borrow an acceptance criteria from Simulated Annealing. A solution x is always accepted if $c(x) > c(x_{\text{best}})$. If $c(x) < c(x_{\text{best}})$ then x is accepted with probability $\exp\left(-\frac{c(x_{\text{best}}) - c(x)}{T}\right)$. In each iteration the temperature T is updated by $T = d_{\text{SA}} T$, where $0 < d_{\text{SA}} < 1$. Giving the *temperature control parameter* w_{SA}, $0 < w_{\text{SA}} < 1$, T is initially selected such that a solution is accepted with probability $\frac{1}{2}$ if its change in objective is w_{SA} percent worse than the initial solution x_0, i.e.

$$\exp\left(-\frac{(c(x_0) - (1 - w_{\text{SA}}) \cdot c(x_0))}{T_0}\right) = \tfrac{1}{2} \quad \Rightarrow \quad T_0 = \frac{w_{\text{SA}} \cdot c(x_0)}{\ln(2)} \tag{8.4.11}$$

This has the advantage of better adapting the temperature to each dataset.

Furthermore at the start of each segment (those of the ALNS scoring scheme), the current solution is set to the current best.

8.5 Parameter Tuning

The proposed heuristic contains many free parameters. It is essential that these parameters are tuned to achieve good performance, see e.g. Diao et al. (2003) and Adenso-Diaz and Laguna (2006). Tuning of metaheuristics is usually done by rules-of-thumb and the researchers personal experience. However some well performing automated algorithms have lately been introduced, mainly *ParamILS* (Hutter et al. (2009)) and *Race*-algorithms (Birattari (2005)). In this paper we will use the *F-Race* algorithm for tuning, as its implementation burden seems light, and it has proven competitive for some heuristic methods, see Montero et al. (2010) and Pellegrini et al. (2010).

The main idea of a race algorithm is to sequentially process a set of data instances using all possible parameter configurations. In each iteration, the parameter configurations which are statistically inferior are eliminated. The algorithm is ran until one parameter configuration remains or the specified time limit is exceeded, see Algorithm 1. If more than one parameter configuration remains once the algorithm terminates, the one which in average has performed best is selected. The advantage of a race algorithm is that bad parameter configurations are eliminated early, such that no more valuable computation time is spend on evaluating these. The racing algorithm differs from most other tuning approaches in the sense that it only performs one algorithm run per parameter configuration per data instances. This relies on the proof in Birattari (2005) where it is shown that this is the optimal experimental setting in terms of variance of the estimated performance.

Algorithm 1: Race Tuning

Input:
Θ: Set of parameter configurations
T_{exp}: Computation time of each experiment
T_{total}: Time limit
α: Confidence level

```
1 i = 0, S_θ = Θ, C_θ = ∅
2 while t < T_total AND |S_θ| > 1 do
3 |   dataset = RandomSampled()
4 |   foreach θ ∈ S_θ do
5 |   |   C_θ^i = EvalSolution(T_exp, θ, dataset)
6 |   i = i + 1
7 |   Drop inferior parameter configurations from S_θ by statistical test, using confidence level α
```

In a F-Race algorithm, the *Friedman Two-way Analysis of Variance by Ranks* test is used to determine whether there is sufficient statistically evidence to eliminate parameter configurations from future iterations. If this is the case, then post-tests are performed where pairwise comparison between the best candidate and the remaining determines which configurations should be eliminated, if any. The F-Race algorithm has been successfully used for tuning in a number of cases, see e.g. Becker et al. (2006); Chiarandini et al. (2006); Pellegrini and Birattari (2007).

A problem of the described Racing algorithm, which applies to most tuning frameworks, is the so-called full-factorial design, meaning that the full set of parameter configuration is initially considered. This results in the F-Race becoming impractical and computational prohibitive, if there exists a large number of parameters and each parameters can take a modest number of values. This has been addressed in Balaprakash et al. (2007) by defining a probabilistic model on the set of all possible parameter configurations, such that a small set of parameter configurations is generated in each iteration of the tuning process. Elite configurations are used to update the

model to bias the search around high quality parameter configurations. This version of F-Race is denoted *Iterative F-Race* (I/F-Race).

In this paper we use a simplified I/F-Race algorithm, where we start out with a small subset of parameter configurations, and based on the Race-results of these we manually construct new configurations, which are believed to be superior. One could think of this approach as a sort of manual iterative F-Race. Table 8.2 shows the best found parameter configuration. It should be mentioned that we set $b_{\text{SP}} = b_0$ and $\delta^s = \delta^t = \infty$, since these are the most common values chosen by the users. The datasets used are of the school year 2011/2012. w_{SA} is the temperature control

Table 8.2: Final values of tuned parameters, found by the F-Race algorithm with confidence level $\alpha = 0.05$.

Parameter	w_{SA}	d_{SA}	N_{it}	ρ	σ	ξ_{start}	ξ_{end}	p_{shaw}
Value	0.01	0.99	100	0.50	1000	0.30	0.0033	20

parameter and d_{SA} is the decay parameter for the SA based acceptance criteria. N_{it} defines the number of iterations between resets. ρ and σ are reaction factor and the scale parameter for the ALNS scoring scheme, respectively. ξ_{start} and ξ_{end} are the destroy percentage in the beginning and in the end of the running time. Finally, p_{shaw} indicates how random the element is chosen in the Shaw removal.

8.6 Performance

The goal of this section is to evaluate the performance of the developed solution methods, the ALNS algorithm and solving the IP model. Also a comparison with the existing heuristic of Lectio is made. All tests are performed using nUnit 2.6 in C# 4.0 on a machine with an Intel i7-930@2.8GHz CPU and 12GB of RAM. No parallelization has been implemented.

8.6.1 Performance comparison between ALNS and Gurobi

In the following, the performance of the state-of-the-art MIP solver Gurobi 5.01 (currently top-ranked in the MIP benchmark of Mittelman (2013)) and the implemented ALNS algorithm are compared. For both the PCTP and the SCTP, 100 datasets from the school-year 2011/2012 are selected from the database of Lectio.

In this experimental setup, the ALNS algorithm is run for 2 minutes. This low running time is due to the following: 1) The schools does generally not expect an algorithm to run longer, as they are usually not aware that it is a hard problem to solve. Some even believe the problem is trivial. 2) The ALNS "tailors-off" after a while, i.e only minor improvements are seen on the best found solution after the 2 minute mark.

The Gurobi solver is run for 1 hour, because we do not only want to evaluate the performance in terms of best found IP solution, we also want a good upper bound for the instances.

In the performance tests it is not allowed to interrupt other activities for the PCTP, i.e. $\delta_t = \delta_s = \infty$. For the SCTP interrupting activities is allowed. This is due to the fact that PCTP is normally arranged in the evening while SCTP is during the normal work-hours.

From Table 8.1 it is seen that ALNS in average finds solutions 4% from optimum. Even though ALNS has far lower running time than Gurobi, it finds better solutions in almost all cases.

Table 8.1: Gurobi 5.01 and ALNS for the PCTP on 100 datasets. For each dataset is listed the number of time slots "$|B|$", the total number of meeting requests "$\sum R$", the average number of requests pr. student, and the average number of requests pr. teacher. For Gurobi is shown the final objective value "x", the best bound found "UB", and the reported gap between these two. For ALNS, column "$\bar{x}$" is the mean performance of the algorithm over 10 runs, and column "σ" is the standard deviation for these runs. Finally, column "Gap" is the gap of mean performance and the upper bound found by Gurobi. The best found solution is marked with bold for each instance.

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Alleroed	12	51	3.0	2.4	**485.0**	485.0	0.0	484.8	0.1	0.0
Alssund	18	84	2.5	4.0	**850.6**	850.6	0.0	849.4	0.6	0.1
Aurehoej1	18	537	4.0	9.9	3270.3	3774.3	15.4	**3655.9**	6.9	3.2
Aurehoej2	18	409	3.8	6.6	3033.1	3300.5	8.8	**3219.5**	3.9	2.5
Broenderslev	24	241	4.0	4.8	1656.2	1965.1	18.7	**1905.6**	3.0	3.1
CPHWEST	39	133	3.7	5.8	990.5	1124.7	13.5	**1044.5**	3.0	7.7
DetKristne	32	247	4.1	11.2	1553.3	1978.5	27.4	**1830.9**	4.9	8.1
Dronninglund1	30	108	4.9	7.7	726.0	801.5	10.4	**782.0**	4.1	2.5
Dronninglund2	30	94	4.1	8.6	636.6	672.7	5.7	**664.5**	0.6	1.2
Egedal	27	408	3.4	6.3	3143.5	3625.0	15.3	**3558.2**	3.7	1.9
Egaa	24	265	3.2	9.8	2101.7	2374.0	13.0	**2318.5**	7.7	2.4
Esbjerg1	24	345	3.9	9.6	2306.2	2465.3	6.9	**2402.1**	4.2	2.6
Esbjerg2	24	307	3.5	4.3	2117.6	2439.9	15.2	**2314.3**	2.4	5.4
Esbjerg3	24	255	3.8	9.4	1770.0	1888.1	6.7	**1839.5**	2.8	2.6
Esbjerg4	24	351	4.0	9.2	2493.8	2700.5	8.3	**2612.3**	4.5	3.4
Frederikssund	24	49	3.3	4.5	**404.2**	406.8	0.7	403.8	0.4	0.8
Frederiksvaerk	8	74	2.4	3.2	**699.0**	699.0	0.0	697.9	0.6	0.2
Gefion	18	479	3.1	7.2	3316.6	4248.9	28.1	**3958.1**	23.7	7.4
Gladsaxe	40	901	4.1	11.0	5516.0	7163.7	29.9	**6950.7**	15.1	3.1
Greve	18	336	4.5	4.4	2133.1	2535.9	18.9	**2482.6**	4.4	2.2
Haslev1	18	123	2.9	5.4	1051.5	1077.1	2.4	**1060.2**	0.5	1.6
Haslev2	18	122	3.2	4.2	983.5	1019.6	3.7	**988.7**	2.6	3.1
Herlufsholm1	24	143	4.3	14.3	853.1	918.9	7.7	**894.5**	1.2	2.7
Herlufsholm2	24	88	4.9	6.8	599.5	671.3	12.0	**621.8**	1.9	8.0
Herning1	27	118	3.8	3.6	0.0	0.0	0.0	0.0	0.0	0.0
Herning2	27	75	3.4	6.3	0.0	0.0	0.0	0.0	0.0	0.0
Herning3	27	140	3.7	4.5	2.0	2.0	0.0	2.0	0.0	0.0
Himmelev	34	453	3.7	8.4	3118.1	3736.1	19.8	**3471.7**	8.8	7.6
Hjoerring	30	179	3.8	2.1	751.2	1229.9	63.7	**1009.2**	15.3	21.9
HorsensGym	18	123	3.2	3.4	**1131.5**	1133.8	0.2	1129.4	1.1	0.4
HorsensStats	21	143	2.9	3.0	1217.8	1289.3	5.9	**1253.8**	1.9	2.8
Ikast-Brande	30	52	3.5	5.8	419.4	463.7	10.6	**449.7**	0.4	3.1
Johannesskolen1	24	165	4.5	7.9	1131.2	1296.8	14.6	**1188.3**	4.3	9.1
Johannesskolen2	24	97	3.7	7.5	701.4	786.3	12.1	**743.9**	5.0	5.7
Johannesskolen3	28	135	3.6	7.5	519.2	581.4	12.0	**519.9**	3.2	11.8
Kalundborg	27	299	3.0	4.8	2281.0	2587.0	13.4	**2458.3**	5.3	5.2
Kolding	18	80	3.5	4.0	**721.4**	725.2	0.5	721.0	0.2	0.6
Langkær1	18	52	3.5	2.6	**471.1**	471.1	0.0	470.6	0.3	0.1
Langkær2	18	90	3.5	3.8	788.9	814.6	3.3	**805.9**	0.5	1.1
Middelfart	27	223	3.1	6.2	1682.5	1923.6	14.3	**1788.1**	7.4	7.6
Morsoe1	27	105	3.9	10.5	768.8	834.2	8.5	**804.2**	3.2	3.7
Morsoe2	27	113	4.7	10.3	732.9	815.2	11.2	**778.4**	4.2	4.7

Continued on next page

Table 8.1 – continued from previous page

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Munkensdam1	21	256	3.6	4.7	1871.9	2251.6	20.3	**2198.6**	5.3	2.4
Munkensdam2	21	345	3.4	6.3	2352.8	2931.6	24.6	**2846.8**	6.7	3.0
NielsSteensens1	36	117	5.9	7.8	715.8	781.9	9.2	**757.2**	3.9	3.3
NielsSteensens2	30	328	7.5	17.3	1249.9	1764.9	41.2	**1656.6**	5.9	6.5
NielsSteensens3	30	365	7.9	20.3	1125.7	1895.7	68.4	**1800.3**	7.4	5.3
NielsSteensens4	30	234	6.7	13.8	1101.6	1230.0	11.7	**1144.4**	5.2	7.5
NielsSteensens5	30	263	6.3	17.5	1461.7	1634.2	11.8	**1557.1**	5.7	5.0
Noerre	18	422	2.7	7.8	3574.1	4031.6	12.8	**3944.5**	5.2	2.2
Nordfyns	23	192	2.6	6.6	1761.9	1863.5	5.8	**1795.5**	6.2	3.8
Nordsjaellands1	34	1187	6.4	25.3	6001.7	7018.5	16.9	**6597.3**	27.9	6.4
Nordsjaellands2	34	1038	6.6	23.1	2298.7	2626.1	14.3	**2453.2**	14.7	7.1
Nordsjaellands3	34	457	6.3	13.9	2225.4	2858.3	28.4	**2634.7**	7.8	8.5
Nordsjaellands4	34	163	4.9	9.6	1100.9	1210.0	9.9	**1172.4**	2.9	3.2
Nordsjaellands5	40	712	5.6	19.8	2543.1	4796.2	88.6	**4460.5**	16.6	7.5
Nordsjaellands6	34	780	6.1	19.0	2500.4	4899.0	95.9	**4612.2**	14.2	6.2
Nordsjaellands7	34	880	6.1	19.1	2763.3	3047.9	10.3	**2894.5**	6.3	5.3
Nordsjaellands8	34	23	1.6	3.3	**242.1**	242.1	0.0	241.9	0.3	0.1
Nordsjaellands9	34	949	6.2	22.1	3202.9	5519.0	72.3	**5037.1**	31.8	9.6
Nordsjaellands10	34	31	1.9	4.4	**272.2**	272.9	0.2	269.1	3.3	1.4
Nyborg	24	119	3.2	5.7	55.4	55.4	0.0	55.4	0.0	0.0
Nykoebing	24	182	3.0	3.2	1447.1	1502.2	3.8	**1483.1**	1.2	1.3
NZahles1	25	324	4.3	9.5	2116.2	2456.3	16.1	**2365.7**	5.5	3.8
NZahles2	24	301	4.1	8.9	2035.3	2280.1	12.0	**2217.8**	6.5	2.8
Odder	18	95	4.0	7.3	740.0	773.5	4.5	**762.7**	1.5	1.4
Odsherreds	21	193	3.4	5.0	1533.4	1619.6	5.6	**1595.4**	2.2	1.5
Risskov1	15	65	3.1	4.6	**539.7**	539.7	0.0	536.7	2.5	0.6
Risskov2	15	149	3.5	5.7	**1263.2**	1273.1	0.8	1256.9	2.9	1.3
Risskov3	15	181	3.7	6.2	**1396.9**	1406.8	0.7	1389.7	4.8	1.2
Roedkilde	18	266	3.6	9.2	2161.1	2352.9	8.9	**2325.2**	5.2	1.2
Roedovre	51	779	3.6	10.3	1513.8	2032.4	34.3	**1661.7**	20.1	22.3
Rosborg1	24	218	3.5	9.9	1742.7	1876.3	7.7	**1827.9**	5.4	2.7
Rosborg2	28	268	3.7	11.7	1960.9	2297.2	17.2	**2223.3**	11.4	3.3
Rosborg3	28	487	1.9	9.6	4568.5	4939.6	8.1	**4750.0**	10.1	4.0
Rosborg4	26	235	3.6	8.4	1713.8	2033.2	18.6	**1960.4**	11.1	3.7
Roskilde	48	263	3.7	6.4	1716.4	2251.7	31.2	**2112.8**	6.4	6.6
Rybners	24	267	3.1	6.9	1923.3	2472.3	28.5	**2402.5**	3.8	2.9
SanktAnnae	21	320	4.1	7.0	2115.0	2498.7	18.1	**2381.6**	5.6	4.9
Skive	36	220	3.6	12.9	1611.0	1903.2	18.1	**1850.3**	2.9	2.9
Slagelse	30	85	3.0	3.9	**805.3**	805.4	0.0	802.6	0.8	0.4
Solroed1	16	341	3.3	7.4	2418.2	2468.7	2.1	**2436.8**	6.1	1.3
Solroed2	16	415	3.4	7.2	3083.3	3317.9	7.6	**3263.0**	5.2	1.7
Soroe	24	369	3.7	8.2	2587.1	3111.2	20.3	**2947.3**	12.1	5.6
Soroe	33	335	4.2	5.6	1594.3	2649.3	66.2	**2255.0**	12.3	17.5
Stenhus	18	221	4.3	3.2	0.0	0.0	0.0	0.0	0.0	0.0
Stoevring	24	62	3.7	4.4	**521.4**	521.4	0.0	520.5	0.5	0.2
Struer1	30	237	3.3	4.4	1596.8	1801.2	12.8	**1656.9**	2.1	8.7
Struer2	30	333	3.3	8.8	2301.3	2790.8	21.3	**2534.0**	7.9	10.1
Svendborg1	18	96	2.1	2.9	**991.4**	991.4	0.0	991.1	0.2	0.0
Svendborg2	18	134	2.6	4.5	**1289.0**	1289.1	0.0	1288.0	0.3	0.1

Continued on next page

Table 8.1 – continued from previous page

					Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|S\|}$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Taarnby	36	791	4.6	11.0	4397.7	5918.0	34.6	**5609.2**	17.7	5.5
UCH	32	104	1.0	17.3	**949.8**	949.8	0.0	922.6	0.0	2.9
ViborgGym1	30	206	4.3	5.2	1367.7	1486.5	8.7	**1434.0**	2.6	3.7
ViborgGym2	30	149	4.4	5.3	1094.0	1146.6	4.8	**1133.4**	0.9	1.2
ViborgGym3	30	294	3.7	4.6	2153.6	2275.5	5.7	**2211.7**	1.8	2.9
ViborgHandel	30	324	4.2	18.0	2111.8	2615.9	23.9	**2526.7**	9.7	3.5
ViborgKatedral	40	337	4.8	11.2	1935.5	2516.5	30.0	**2313.4**	8.6	8.8
Vordingborg1	16	315	3.8	6.3	2222.0	2358.8	6.2	**2304.3**	2.8	2.4
Vordingborg2	16	239	3.3	5.6	1867.5	1950.0	4.4	**1924.6**	2.0	1.3
Average	26	279	3.9	8.3			14.8			4.0
Max	51	1187	7.9	25.3			95.9			22.3

Table 8.2 shows the performance for the SCTP.

Table 8.2: Gurobi and ALNS for the SCTP on 100 datasets. Columns are defined in analogous way to Table 8.1. The average number of requests pr. student is not shown, as it is 1.0 in all cases.

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Aabenraa	60	226	4.3	**2461.2**	2495.7	1.4	2387.5	3.3	4.5
Broendby1	21	69	3.8	677.3	707.3	4.4	**683.2**	2.8	3.5
Broendby2	14	69	4.3	**770.2**	779.5	1.2	768.6	1.0	1.4
Broendby3	24	62	3.4	609.0	632.8	3.9	**614.8**	1.3	2.9
Broenderslev1	69	115	3.5	**1335.4**	1340.9	0.4	1302.4	2.1	3.0
Broenderslev2	102	115	3.5	**1253.4**	1272.5	1.5	1236.0	1.6	3.0
Christianshavns	43	210	4.8	**2329.7**	2423.7	4.0	2223.7	4.4	9.0
Dronninglund1	100	134	4.2	1442.7	1481.3	2.7	**1453.5**	2.1	1.9
Dronninglund2	60	134	4.3	**1553.3**	1561.9	0.6	1537.5	2.8	1.6
Egaa	29	214	4.8	2257.5	2457.1	8.8	**2376.2**	4.0	3.4
Falkoner1	30	64	3.4	**671.6**	679.0	1.1	668.0	2.6	1.6
Falkoner2	37	206	4.2	2188.3	2345.6	7.2	**2266.1**	5.0	3.5
Falkoner3	30	64	3.4	664.9	672.3	1.1	664.9	0.0	1.1
Grenaa1	28	122	3.8	1280.7	1380.3	7.8	**1325.4**	3.1	4.2
Grenaa2	24	122	3.8	1249.2	1330.7	6.5	**1290.3**	3.8	3.1
Greve1	28	157	3.3	1575.1	1762.2	11.9	**1693.3**	3.2	4.1
Greve2	62	259	4.1	2805.4	3035.8	8.2	**2913.5**	7.4	4.2
Greve3	20	51	3.2	**566.4**	569.9	0.6	566.3	0.0	0.6
Gribskov1	74	182	4.1	**1867.8**	1914.6	2.5	1787.6	3.2	7.1
Herlev1	24	71	2.8	729.9	751.5	3.0	**730.2**	0.9	2.9
Herlev2	29	78	2.8	682.1	791.7	16.1	**750.2**	1.2	5.5
Hoeng1	21	66	3.5	607.9	686.5	12.9	**621.5**	0.4	10.5
Hoeng2	23	98	3.9	1029.8	1071.8	4.1	**1038.5**	2.2	3.2
Hoeng3	22	45	2.7	392.7	481.7	22.7	**408.3**	1.2	18.0
Hoeng4	23	56	2.6	589.1	612.4	4.0	**589.2**	1.8	3.9
Koebenhavns1	16	143	3.9	1239.8	1278.3	3.1	**1242.0**	6.1	2.9
Koebenhavns2	16	100	3.7	**786.4**	786.5	0.0	785.6	0.8	0.1
Koebenhavns3	16	100	3.7	725.8	725.9	0.0	725.8	0.0	0.0
Koebenhavns4	25	146	3.8	1402.3	1486.1	6.0	**1424.6**	0.8	4.3

Continued on next page

Table 8.2 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Koege1	30	255	8.5	1348.2	2475.2	83.6	**2333.2**	10.3	6.1
Koege2	36	261	6.2	**2381.3**	2493.7	4.7	2045.7	5.2	21.9
Koege3	74	258	8.6	**2775.1**	2903.2	4.6	2622.9	5.0	10.7
Kolding1	24	219	5.0	2283.5	2430.9	6.5	**2348.7**	4.8	3.5
Kolding2	45	174	3.8	**1934.9**	2005.3	3.6	1908.8	4.1	5.1
Langkaer1	62	215	5.4	2195.8	2472.6	12.6	**2239.4**	7.8	10.4
Langkaer2	60	216	5.4	**2351.0**	2481.0	5.5	2240.1	9.3	10.8
Langkaer3	60	216	5.4	**2359.1**	2472.8	4.8	2258.3	6.6	9.5
Langkaer4	30	57	3.8	546.4	596.1	9.1	**566.2**	3.6	5.3
Langkaer5	56	217	5.6	**2282.3**	2493.5	9.3	2253.3	3.4	10.7
Langkaer6	62	56	3.7	**631.5**	652.6	3.3	629.2	0.7	3.7
Mariagerfjord1	29	123	4.0	1227.0	1387.2	13.1	**1318.6**	2.0	5.2
Mariagerfjord2	29	123	4.0	1269.0	1402.6	10.5	**1345.3**	4.0	4.3
Marselisborg1	22	102	3.4	1021.3	1087.4	6.5	**1045.4**	1.9	4.0
Marselisborg2	17	106	3.3	**1036.2**	1046.9	1.0	1035.4	1.0	1.1
Marselisborg3	22	105	3.9	1049.3	1156.3	10.2	**1098.3**	4.3	5.3
Marselisborg4	17	96	3.2	**948.6**	955.2	0.7	947.2	0.9	0.8
Munkensdam	43	191	5.6	**2179.9**	2203.8	1.1	2067.9	4.6	6.6
Nordfyns1	22	173	5.1	1871.9	1974.3	5.5	**1926.6**	1.6	2.5
Nordfyns2	21	173	5.2	1846.9	1975.6	7.0	**1929.9**	2.6	2.4
Nordfyns3	22	173	5.1	1831.5	1973.7	7.8	**1908.1**	2.1	3.4
Nordfyns4	21	173	4.1	1438.0	1538.2	7.0	**1478.3**	3.7	4.1
Noerresundby	31	303	4.7	2959.5	3437.4	16.2	**3291.7**	2.6	4.4
NZahles1	13	69	3.3	615.7	636.9	3.5	**619.1**	0.9	2.9
NZahles2	13	62	3.9	**512.1**	524.3	2.4	509.5	1.3	2.9
Odsherreds	49	119	3.8	**1365.1**	1372.6	0.6	1289.4	3.4	6.5
Oeregaard1	20	219	5.6	2257.3	2296.6	1.7	**2258.1**	5.6	1.7
Oeregaard2	20	213	5.0	1728.9	1778.5	2.9	**1743.9**	3.2	2.0
Oeregaard3	20	219	5.6	2327.5	2372.3	1.9	**2340.2**	3.3	1.4
Oeregaard4	20	219	5.6	2338.4	2373.5	1.5	**2339.9**	2.6	1.4
Risskov	36	215	6.1	**2400.4**	2427.4	1.1	2353.2	2.0	3.2
Roedkilde	18	230	4.4	2452.3	2534.1	3.3	**2495.7**	3.1	1.5
Rosborg1	22	257	4.9	2756.7	2895.6	5.0	**2837.4**	3.2	2.1
Rosborg2	22	257	4.8	2651.0	2860.6	7.9	**2805.4**	3.3	2.0
SanktAnnae1	23	149	3.6	1554.1	1675.1	7.8	**1580.4**	3.9	6.0
SanktAnnae2	24	165	3.8	1682.9	1850.2	9.9	**1753.4**	4.0	5.5
SanktAnnae3	17	21	2.6	197.9	197.9	0.0	197.9	0.0	0.0
SanktAnnae4	31	162	4.2	1359.0	1719.9	26.6	**1598.9**	3.6	7.6
Skanderborg1	57	232	3.9	2440.0	2640.1	8.2	**2547.4**	3.4	3.6
Skanderborg2	60	229	4.9	**2369.5**	2414.4	1.9	2320.4	3.4	4.1
Skive1	16	140	3.3	1417.5	1458.6	2.9	**1430.9**	3.8	1.9
Skive2	31	103	2.6	865.5	1062.6	22.8	**995.5**	2.5	6.8
Skive3	31	140	3.3	1189.7	1452.9	22.1	**1372.7**	4.4	5.8
Skive4	16	21	2.1	**227.8**	227.8	0.0	227.6	0.1	0.1
Skive5	16	98	3.0	**960.9**	972.9	1.3	960.7	0.9	1.3
Skive6	16	110	3.1	1106.9	1144.7	3.4	**1119.3**	2.0	2.3
Skive7	31	134	3.0	1152.9	1365.6	18.5	**1284.6**	3.2	6.3
Skive8	16	107	3.0	1006.8	1015.8	0.9	**1007.1**	0.9	0.9
Skive9	31	100	3.0	907.8	1034.9	14.0	**983.0**	4.6	5.3

Continued on next page

Table 8.2 – continued from previous page

				Gurobi 5.01			ALNS		
	$\|B\|$	$\sum R$	$\frac{\sum R}{\|T\|}$	x	UB	Gap[%]	$\bar{x}$	σ	Gap[%]
Soenderborg1	22	234	3.7	2105.1	2590.6	23.1	**2475.3**	4.9	4.7
Soenderborg2	22	236	3.4	2095.1	2703.9	29.1	**2577.9**	4.2	4.9
Soenderborg3	21	236	3.4	2452.5	2688.1	9.6	**2597.3**	4.0	3.5
Soenderborg4	22	235	3.4	1927.6	2681.3	39.1	**2554.2**	3.7	5.0
Solroed1	18	242	4.6	1935.5	2181.4	12.7	**2130.3**	6.8	2.4
Solroed2	20	22	1.8	228.6	228.6	0.0	228.6	0.0	0.0
Solroed3	17	22	1.8	223.3	223.3	0.0	223.3	0.0	0.0
Solroed4	54	243	4.6	2309.9	2562.4	10.9	**2354.2**	5.2	8.8
Solroed5	20	243	4.5	1555.0	2185.8	40.6	**2054.9**	8.0	6.4
Solroed6	17	215	4.2	1703.0	1835.0	7.8	**1775.5**	6.8	3.4
Solroed7	17	194	4.0	1599.6	1748.0	9.3	**1679.2**	2.6	4.1
Vejen1	10	41	2.4	424.2	424.2	0.0	424.2	0.0	0.0
Vejen2	19	126	3.8	1171.2	1225.0	4.6	**1198.5**	3.0	2.2
Vejen3	19	125	4.2	1172.1	1234.9	5.4	**1204.6**	2.4	2.5
Vejen4	19	125	4.2	1130.2	1205.3	6.6	**1172.5**	2.4	2.8
Viborg1	19	105	3.8	1004.3	1100.3	9.6	**1034.2**	0.6	6.4
Viborg2	49	187	4.4	**2081.7**	2153.0	3.4	2060.0	4.8	4.5
Viby1	20	124	3.7	**1278.3**	1279.4	0.1	1256.9	1.2	1.8
Viby2	13	93	5.2	957.5	957.5	0.0	957.5	0.0	0.0
Viby3	8	45	2.7	480.0	480.0	0.0	480.0	0.0	0.0
Viby4	16	93	5.2	**1057.7**	1057.7	0.0	1053.3	0.3	0.4
Viby5	21	123	3.7	**1378.6**	1378.8	0.0	1356.1	1.1	1.7
Average	30	148	4.1			7.7			4.1
Max	102	303	8.6			83.6			21.9

From Table 8.2 it is seen that ALNS in average finds solutions 4.1% from optimum for the SCTP. This is lower than the average gap for Gurobi, which is 7.7%.

From Table 8.1 it is seen that ALNS outperforms Gurobi for the PCTP. For the SCTP, the results are more blurred, but the ALNS still performs best in 70 out of 100 cases. What can also be seen from Table 8.1 and 8.2 is that the standard deviation for the ALNS is low in all cases, and the maximum gap obtained across all datasets is considerably lower than the maximum gap which Gurobi obtains (even given the higher running time of Gurobi). This is important as the customers of Lectio expects a consistent and stable solution procedure.

8.6.2 Performance comparison of ALNS and current heuristic of Lectio

The current algorithm in Lectio is an undocumented heuristic, which initially attempts to fulfill every meeting request by assigning them to random time slots, and then attempts to find improving solutions with a hill-climber embedded in a Simulated Annealing (SA) framework. This heuristic does not support the SCTP. In this section, we compare the existing heuristic of Lectio with the implemented ALNS algorithm.

The comparison of algorithms for the PCTP is done by adapting the objective of the ALNS so it matches the one of the implemented SA algorithm, which yields the following changes:

- Since the SA algorithm attempts to fulfill all meeting requests, we set α_g^s to a huge value for all meeting requests, effectively making the ALNS behave the same way.
- We set $\beta^t = \gamma^s = 2$.
- The SA algorithm does not allow interrupting of activities, i.e. $\delta^t = \delta^s = \infty$.

- The time slot set-point setting of the SA solver is broken, so we set $\kappa = 0$ and likewise for the SA solver.
- The violation of sequence length for teachers is penalized in quadratic way. This means that term (8.3.21) is now written as $-\sum_{t,b,d}(y_{t,b,d})^{\omega}$, and $\omega = 2$.

We evaluate the algorithms on 100 randomly selected datasets for the school-year 2009/2010. The reason a new batch of datasets is selected for this test is that the existing heuristic of Lectio does not support all features mentioned in this paper. Due to customer requests, Lectio is continuously developed, and this also effects the CTP. E.g. datasets from the school-year 2009/2010 does not support features such as multiple days for a consultation.

Experience has shown that the SA algorithm needs long runtime to provide meaningful solutions. We set runtime equal to 10 minutes, which is significantly higher than the preferable runtime of the high schools, as described in Section 8.6.1. Furthermore, to reduce the influence of stochastic behavior, we perform 10 runs on each dataset with each solver.

Table 8.3 shows the average performance, the standard deviation and the number of unassigned requests of both algorithms. Recall that in this test both algorithms attempt to fulfill every meeting request. However it cannot be guaranteed that the algorithms can find a solution which satisfies this, nor that it even exists. Furthermore we compare the algorithms in the domain of the SA algorithm, and in this domain it is only attempted to minimize the different costs, i.e. no benefit is made from fulfilling meeting requests. This means it would not be fair to compare solutions which does not have the same amount of unassigned requests, since additional fulfilled requests will potentially yield additional cost of e.g. number of breaks, interrupted activities, etc. Therefore we enforce the following criterion to determine whether the comparison of solutions for a dataset is valid: The difference in the number of unassigned requests must lie in the interval ± 1. This means the comparison of solvers in Table 8.3 is only approximate, however it can be considered as a very good approximation, since a difference of one fulfilled meeting request will have minor influence of the objective. Notice that the objectives are given in the domain of the SA algorithm, which is of different magnitude than the objective of this article. This is due to the fact that the undocumented heuristic aims at minimizing whereas the approach of this paper is to maximize.

Given the average objective of the SA algorithm $\bar{x}_{\text{SA}}$ and the average objective of the ALNS algorithm $\bar{x}_{\text{ALNS}}$, and that the comparison of these two is valid, we compute the difference $\bar{x}_{\text{SA}} - \bar{x}_{\text{ALNS}}$. For almost every instance where comparison is valid, the ALNS algorithm in average finds a better solution. Furthermore the solutions from the ALNS algorithm has far lower deviation than the SA algorithm, which are important, as the users of the algorithm will usually only run the algorithm once.

Table 8.3: Comparison of performance of the SA algorithm and the ALNS algorithm. Each algorithm is ran 10 times on each dataset. For each algorithm and each dataset is listed the mean objective "$\bar{x}$", standard deviation of objective "σ", and the number of unassigned meeting requests "#UA". Notice that the objectives are given in the domain of the SA algorithm, which is of different magnitude than the objective of this article. Those datasets where the number of unassigned requests differs between the algorithms with more than ± 1 are struck out, as this is not considered a fair comparison. Column "Diff" is the difference between mean objectives.

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Aabenraa1	640.90	83.36	17.0	**555.00**	0.00	17.0	85.90
Aabenraa2	384.00	41.86	9.0	**315.90**	18.27	9.0	68.10
~~Aabenraa3~~	~~41.60~~	~~11.27~~	~~12.4~~	~~123.90~~	~~7.61~~	~~9.0~~	~~N/A~~
~~Aabenraa4~~	~~126.90~~	~~59.36~~	~~13.5~~	~~166.80~~	~~53.45~~	~~12.0~~	~~N/A~~
Aabenraa5	**36.00**	9.30	0.0	37.60	0.84	0.0	-1.60
Aabenraa6	0.00	0.00	60.0	0.00	0.00	60.0	0.00

Continued on next page

Table 8.3 – continued from previous page

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Aabenraa7	0.00	0.00	52.0	0.00	0.00	52.0	0.00
Aurehoej	283.30	37.99	12.8	**165.70**	12.72	12.0	117.60
~~Birkeroed1~~	~~542.10~~	~~88.05~~	~~40.2~~	~~171.70~~	~~37.57~~	~~35.5~~	~~N/A~~
Birkeroed2	493.80	88.41	30.1	**92.40**	19.21	29.5	401.40
Bjerringbro	2994.60	390.30	22.0	**1393.20**	16.90	22.0	1601.40
~~Borupgaard1~~	~~36.60~~	~~11.66~~	~~10.9~~	~~176.70~~	~~11.98~~	~~2.0~~	~~N/A~~
~~Borupgaard2~~	~~16.00~~	~~8.03~~	~~51.2~~	~~221.40~~	~~1.71~~	~~41.0~~	~~N/A~~
~~Borupgaard3~~	~~32.80~~	~~23.67~~	~~26.3~~	~~115.80~~	~~10.34~~	~~20.0~~	~~N/A~~
Broenderslev	154.40	51.04	0.0	**22.20**	6.96	0.0	132.20
CPHWEST	1244.40	38.56	0.0	**1105.00**	0.00	0.0	139.40
Esbjerg	0.00	0.00	407.0	0.00	0.00	407.0	0.00
Fjerritslev	2088.50	72.25	0.0	**1904.20**	6.34	0.0	184.30
Frederikssund1	330.40	39.98	0.0	**257.70**	6.34	0.0	72.70
Frederikssund2	436.40	22.85	0.0	**340.00**	4.59	0.0	96.40
Frederiksvaerk	25.90	1.85	34.0	**25.00**	0.00	34.0	0.90
GlHellerup1	130.90	29.37	4.0	**29.50**	5.78	4.0	101.40
GlHellerup2	268.70	36.58	1.0	**88.60**	12.95	1.0	180.10
Gefion	112.40	22.04	0.0	**16.70**	3.65	0.0	95.70
~~Gladsaxe~~	~~85.40~~	~~16.40~~	~~87.8~~	~~1034.00~~	~~37.54~~	~~27.0~~	~~N/A~~
Haderslev1	0.00	0.00	225.0	0.00	0.00	225.0	0.00
Haderslev2	0.00	0.00	185.0	0.00	0.00	185.0	0.00
Haslev1	1239.10	56.96	0.0	**1181.00**	0.00	0.0	58.10
Haslev2	0.00	0.00	51.0	0.00	0.00	51.0	0.00
Hasseris1	10.20	4.52	0.0	**0.90**	0.57	0.0	9.30
Hasseris2	68.80	18.94	0.0	**14.90**	2.56	0.0	53.90
Hasseris3	34.30	25.95	2.9	**5.10**	2.02	3.0	29.20
Hasseris4	11.10	6.23	0.0	**0.10**	0.32	0.0	11.00
Herlufsholm1	46.70	8.90	0.0	**34.30**	5.54	0.0	12.40
Herlufsholm2	86.30	14.35	1.2	**60.20**	4.05	0.0	26.10
~~Herlufsholm3~~	~~2.40~~	~~2.41~~	~~38.4~~	~~0.00~~	~~0.00~~	~~37.0~~	~~N/A~~
~~Herlufsholm4~~	~~267.30~~	~~32.27~~	~~3.3~~	~~176.90~~	~~9.80~~	~~2.0~~	~~N/A~~
Herlufsholm5	15.70	11.86	0.0	**1.60**	1.84	0.0	14.10
Herning1	843.30	119.31	1.0	**632.10**	9.00	1.0	211.20
Herning2	103.50	16.25	0.0	**74.20**	0.42	0.0	29.30
Himmelev	262.10	35.44	0.0	**96.30**	16.22	0.0	165.80
Horsens	17.20	7.50	2.0	**6.50**	1.35	2.0	10.70
Kongsholm	102.20	39.82	21.4	**62.30**	8.64	21.0	39.90
~~Langkaer~~	~~29.20~~	~~8.22~~	~~11.7~~	~~190.90~~	~~10.38~~	~~0.0~~	~~N/A~~
Mariagerfjord	190.00	52.75	0.9	**89.00**	17.58	1.0	101.00
Morsoe1	17.20	9.74	3.0	**0.50**	0.71	3.0	16.70
Morsoe2	44.00	21.91	0.0	**2.70**	1.42	0.0	41.30
Mulernes1	21.50	8.22	0.0	**2.60**	0.97	0.0	18.90
Mulernes2	27.60	12.51	0.0	**1.30**	1.16	0.0	26.30
NZahles1	97.40	28.46	2.2	**64.50**	0.71	2.0	32.90
NZahles2	111.20	35.32	0.9	**72.60**	6.10	0.0	38.60
NielsSteensen1	309.80	42.98	3.0	**214.40**	38.20	3.0	95.40
NielsSteensen2	124.10	24.92	0.0	**90.50**	22.65	0.0	33.60
NielsSteensen3	186.40	37.65	5.0	**147.60**	22.31	5.0	38.80
Nordsjaelland1	1036.30	74.52	4.0	**652.20**	55.28	4.0	384.10

Continued on next page

Table 8.3 – continued from previous page

	Lectio SA			ALNS			
	$\bar{x}$	σ	#UA	$\bar{x}$	σ	#UA	Diff.
Nordsjaelland2	69.50	2.17	586.0	**68.50**	1.58	586.0	1.00
Nordsjaelland3	**1681.40**	132.07	45.0	1927.20	191.16	45.0	-245.80
Nordsjaelland4	**1627.80**	135.70	65.7	2335.90	270.87	65.8	-708.10
Nordsjaelland5	1699.10	118.08	27.6	**1104.40**	76.82	27.0	594.70
Nordsjaelland6	1369.00	102.13	32.1	**1109.90**	61.44	32.0	259.10
Nordsjaelland7	1646.40	106.74	49.1	**1567.70**	87.35	49.0	78.70
Nordsjaelland8	2065.30	179.20	37.0	**1736.00**	93.59	37.0	329.30
Nyborg1	89.70	16.45	0.0	**72.10**	0.32	0.0	17.60
Nyborg2	93.50	23.75	0.0	**51.00**	1.49	0.0	42.50
Nyborg3	0.40	0.70	55.6	**0.00**	0.00	55.0	0.40
Nyborg4	14.40	6.75	3.0	**1.50**	0.85	3.0	12.90
Naerum	424.50	43.07	9.0	**248.40**	50.63	9.0	176.10
Noerresundby1	36.10	18.11	0.0	**9.90**	3.54	0.0	26.20
Noerresundby2	115.70	26.60	0.3	**34.90**	10.12	0.0	80.80
Odder	69.80	27.68	6.0	**21.00**	0.00	6.0	48.80
~~Oure~~	~~40.80~~	~~11.13~~	~~15.7~~	~~24.40~~	~~8.22~~	~~14.0~~	~~N/A~~
Paderup	141.50	15.81	0.0	**75.50**	4.48	0.0	66.00
Randers1	168.40	22.17	0.0	**93.50**	0.53	0.0	74.90
Randers2	244.00	77.90	0.0	**10.80**	3.79	0.0	233.20
Rosborg1	0.00	0.00	246.0	0.00	0.00	246.0	0.00
~~Rosborg2~~	~~2611.00~~	~~299.16~~	~~19.1~~	~~837.10~~	~~75.08~~	~~18.0~~	~~N/A~~
Roskilde	202.30	60.81	0.0	**139.00**	23.25	0.0	63.30
Rybners1	274.90	45.42	0.0	**174.90**	1.10	0.0	100.00
Rybners2	1043.80	67.33	15.0	**789.10**	4.15	15.0	254.70
Roedkilde1	10.90	4.84	0.0	**2.00**	1.33	0.0	8.90
Roedkilde2	19.30	7.48	7.0	**12.00**	2.67	7.0	7.30
Roedkilde3	57.10	11.75	0.0	**6.80**	2.35	0.0	50.30
~~SanktAnnae1~~	~~205.90~~	~~57.43~~	~~60.3~~	~~201.30~~	~~22.07~~	~~51.6~~	~~N/A~~
SanktAnnae2	0.00	0.00	7.0	0.00	0.00	7.0	0.00
Skanderborg1	3.50	1.65	0.0	**0.00**	0.00	0.0	3.50
Skanderborg2	35.40	34.59	0.9	**5.80**	3.79	1.0	29.60
Skive	176.50	47.52	18.0	**89.00**	25.05	18.0	87.50
Slagelse1	130.80	5.94	3.0	**114.90**	5.49	3.0	15.90
Slagelse2	29.70	19.44	0.2	**3.50**	2.55	0.0	26.20
Soroe	2117.10	111.78	14.5	**883.20**	49.02	13.5	1233.90
Stoevring1	17.50	4.93	0.0	**2.70**	1.42	0.0	14.80
Stoevring2	229.20	55.04	0.0	**16.40**	4.14	0.0	212.80
Stoevring3	260.60	71.41	0.0	**14.30**	2.36	0.0	246.30
Stoevring4	124.70	48.48	0.0	**13.80**	2.25	0.0	110.90
Stoevring5	65.90	22.83	6.0	**10.30**	2.54	6.0	55.60
Varde	867.40	152.80	10.0	**326.20**	9.46	10.0	541.20
Vejen	0.30	0.48	0.0	**0.00**	0.00	0.0	0.30
ViborgG	267.90	8.84	94.0	**257.80**	1.03	94.0	10.10
ViborgH	102.20	28.61	0.0	**22.60**	9.16	0.0	79.60
~~Viby~~	~~236.30~~	~~3.65~~	~~186.8~~	~~230.60~~	~~0.52~~	~~181.8~~	~~N/A~~
Average	431.53	47.78	33.9	319.50	20.14	32.5	133.60

By the computational tests of this section, it has been shown that the ALNS algorithm is the best

solution procedure, of those considered in this paper, for the CTP. It outperforms both Gurobi and the existing heuristic of Lectio in terms of both solution quality and reliability.

8.7 Final Remarks and Outlook

It has been shown how the CTP, an important real-life problem for the Danish high schools, can be modeled using linear IP. ALNS has proven successful in establishing solutions for two versions of the problem, the PCTP and the SCTP. Furthermore, F-Race has shown to be an efficient method for tuning of the free parameters. The developed ALNS algorithm has been implemented in Lectio and is hence available for 95% of the Danish high schools.

In case of the PCTP, it has been shown that the ALNS algorithm in average finds solutions which are less than 4% from optimum. This average is taken over 100 real-life dataset, and therefore we have high confidence in this result. Furthermore it has been shown that comparing with the existing algorithm in Lectio, which is the only other known heuristic algorithm for the problem, the ALNS algorithm is far superior. For 83 of the 86 datasets, ALNS finds better solutions, and in many cases the solution quality of the ALNS is considerably better. For the remaining 14 datasets a comparison was not considered fair.

The performance for the SCTP is also tested on 100 real-life dataset. For these datasets, it is shown that the ALNS algorithm in average finds solutions less than 5% from optimum.

For both the PCTP and SCTP the average solution found by ALNS is better than the solutions found by the state-of-the-art MIP solver Gurobi 5.01.

The main subject for further research is considered to be the use of Dantzig-Wolfe decomposition and solution using Branch-and-Price. In this context, a column in the master problem could represent a meeting-plan for a student or a teacher. This would move many constraints to the subproblem, possibly giving a stronger IP formulation, which could lead to a more efficient IP-based solution approach.

Another possibility for future research is to combine the two solution methods described, i.e. using the MIP solver as a repair heuristic within the ALNS. Similar approaches are seen in Prescott-Gagnon et al. (2009) and Muller et al. (2011), with competitive results.

Bibliography

B. Adenso-Diaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

N. Azi, M. Gendreau, and J.-Y. Potvin. *An Adaptive Large Neighborhood Search for a Vehicle Routing Problem with Multiple Trips.* CIRRELT, 2010.

P. Balaprakash, M. Birattari, and T. Stützle. Improvement strategies for the f-race algorithm: sampling design and iterative refinement. In *Proceedings of the 4th international conference on Hybrid metaheuristics*, HM'07, pages 108–122, Berlin, Heidelberg, 2007. Springer-Verlag.

S. Becker, J. Gottlieb, and T. Stützle. Applications of racing algorithms: An industrial perspective. In E.-G. Talbi, P. Liardet, P. Collet, E. Lutton, and M. Schoenauer, editors, *Artificial Evolution*, volume 3871 of *Lecture Notes in Computer Science*, pages 271–283. Springer Berlin / Heidelberg, 2006.

M. Birattari. *The Problem of Tuning Metaheuristics as seen from a Machine Learning Perspective*, volume 292 Dissertations in Artificial Intelligence - Infix. Springer, 1 edition, 2005.

T. Birbas, S. Daskalaki, and E. Housos. School timetabling for quality student and teacher schedules. *J. of Scheduling*, 12:177–197, April 2009. ISSN 1094-6136.

E. Burke and S. Petrovic. Recent research directions in automated timetabling. *European Journal of Operational Research*, 140(2):266 – 280, 2002. ISSN 0377-2217.

M. Carter and G. Laporte. Recent developments in practical course timetabling. In E. Burke and M. Carter, editors, *Practice and Theory of Automated Timetabling II*, volume 1408 of *Lecture Notes in Computer Science*, pages 3–19. Springer Berlin / Heidelberg, 1998.

M. Chiarandini, M. Birattari, K. Socha, and O. Rossi-Doria. An effective hybrid algorithm for university course timetabling. *Journal of Scheduling*, 9:403–432, 2006. ISSN 1094-6136.

P. de Haan, R. Landman, G. Post, and H. Ruizenaar. A case study for timetabling in a dutch secondary school. In E. Burke and H. Rudova, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 267–279. Springer Berlin / Heidelberg, 2007.

Y. Diao, F. Eskesen, S. Froehlich, J. Hellerstein, L. Spainhower, and M. Surendra. Generic online optimization of multiple configuration parameters with application to a database server. In M. Brunner and A. Keller, editors, *Self-Managing Distributed Systems*, volume 2867 of *Lecture Notes in Computer Science*, pages 79–93. Springer Berlin / Heidelberg, 2003.

W. Erben and J. Keppler. A genetic algorithm solving a weekly course-timetabling problem. In E. Burke and P. Ross, editors, *Practice and Theory of Automated Timetabling*, volume 1153 of *Lecture Notes in Computer Science*, pages 198–211. Springer Berlin / Heidelberg, 1996.

F. Hutter, H. Hoos, K. Leyton-Brown, and T. Stützle. Paramils: an automatic algorithm configuration framework. *J. Artif. Int. Res.*, 36:267–306, September 2009. ISSN 1076-9757.

S. Kristiansen and T. R. Stidsen. Adaptive large neighborhood search for student sectioning at danish high schools. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, 2012.

S. Kristiansen, M. Sørensen, and T. R. Stidsen. Elective course planning. *European Journal of Operational Research*, 215(3):713 – 720, 2011. ISSN 0377-2217. doi: 10.1016/j.ejor.2011.06.039.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, June 2013.

G. Laporte, R. Musmanno, and F. Vocaturo. An adaptive large neighbourhood search heuristic for the capacitated arc-routing problem with stochastic demands. *Transportation Science*, 44 (1):125–135, 2010.

H. Lei, G. Laporte, and B. Guo. The capacitated vehicle routing problem with stochastic demands and time windows. *Computers & Operations Research*, 38(12):1775 – 1783, 2011. ISSN 0305-0548. doi: DOI:10.1016/j.cor.2011.02.007.

B. McCollum. University timetabling: Bridging the gap between research and practice. In *in Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 15–35. Springer, 2006.

H. Mittelman. Benchmarks for optimization software, 2013. URL `http://plato.asu.edu/bench.html`.

T. Müller and K. Murray. Comprehensive approach to student sectioning. *Annals of Operations Research*, 181:249–269, 2010. ISSN 0254-5330.

E. Montero, M.-C. Riff, and B. Neveu. An evaluation of off-line calibration techniques for evolutionary algorithms. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, GECCO '10, pages 299–300, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0072-8. doi: 10.1145/1830483.1830540.

L. Muller. An adaptive large neighborhood search algorithm for the resource-constrained project scheduling problem. In *MIC 2009: The VIII Metaheuristics International Conference*, 2009.

L. Muller, S. Spoorendonk, and D. Pisinger. A hybrid adaptive large neighborhood search heuristic for lot-sizing with setup times. *European Journal of Operational Research*, Volume 218(Issue 3):614–623, 2011.

L. F. Muller and S. Spoorendonk. A hybrid adaptive large neighborhood search algorithm applied to a lot-sizing problem. Technical report, DTU Management Engineering, 2010.

P. Pellegrini and M. Birattari. Implementation effort and performance. pages 31–45. 2007.

P. Pellegrini, T. Stützle, and M. Birattari. Off-line vs on-line tuning: A study on max–min ant system for the tsp. In *Swarm Intelligence*, volume 6234 of *Lecture Notes in Computer Science*, pages 239–250. Springer Berlin / Heidelberg, 2010. ISBN 978-3-642-15460-7.

N. Pillay. An overview of school timetabling research. In *Proceedings of the International Conference on the Theory and Practice of Automated Timetabling*, pages 321–335, Belfast, United Kingdom, 2010.

D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Computers & Operations Research*, 34:2403–2435, August 2005. ISSN 0305-0548.

D. Pisinger and S. Ropke. Large neighborhood search. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010. ISBN 978-1-4419-1665-5.

J.-Y. Potvin and J.-M. Rousseau. A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research*, 66(3):331 – 340, 1993. ISSN 0377-2217.

E. Prescott-Gagnon, G. Desaulniers, and L.-M. Rousseau. A branch-and-price-based large neighborhood search algorithm for the vehicle routing problem with time windows. *Networks*, 54(4): 190–204, 2009. ISSN 1097-0037. doi: 10.1002/net.20332.

G. M. Ribeiro and G. Laporte. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, 39(3):728 – 735, 2012. ISSN 0305-0548.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science*, 40:455–472, November 2006. ISSN 1526-5447.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, 194(1):399–412, April 2012. ISSN 0254-5330.

A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13:87–127, 1999. ISSN 0269-2821.

P. Shaw. A new local search algorithm providing high quality solutions to vehicle routing problems, 1997.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In M. Maher and J.-F. Puget, editors, *Principles and Practice of Constraint Programming — CP98*, volume 1520 of *Lecture Notes in Computer Science*, pages 417–431. Springer Berlin / Heidelberg, 1998.

M. Sørensen and T. R. Stidsen. High school timetabling: Modeling and solving a large number of cases in denmark. In *Proceedings of the Ninth International Conference on the Practice and Theory of Automated Timetabling (PATAT 2012)*, pages 359–364. SINTEF, 2012.

A. Tripathy. School timetabling–a case in large binary integer linear programming. *Management Science*, 30(12):1473–1489, 1984.

Chapter 9

A Branch & Price Algorithm for the Generalized Meeting Planning Problem

Niels-Christian Fink Bagger* Matias Sørensen*† Simon Kristiansen*† Thomas R. Stidsen*

*Department of Management Engineering, Technical University of Denmark,
Produktionstorvet, Building 426, DK-2800 Kgs. Lyngby, Denmark
nbag@dtu.dk, msso@dtu.dk, sikr@dtu.dk, thst@dtu.dk

†MaCom A/S
Vesterbrogade 48, 1., DK-1620 Copenhagen V, Denmark

[1]

Abstract Meetings form an important activity in modern society. Persons meet with other people for business or pleasure, but usually a meeting rules out other meetings for the same persons at the same time. Planning a large number of meetings hence present an important but difficult optimization problem.

The Meeting Planning problem comes in all kinds of different variations. In this article, we present a Mixed-Integer Programming model which can model many of these variations and we also present a Branch & Price algorithm which enables optimization of many types of meeting problems. The Branch & Price algorithm is tested on two problems occurring in Danish high schools, the Parental Consultation Timetabling Problem and the Supervisor Consultation Timetabling Problem, each represented by 100 real-world datasets. The tests show that the Branch & Price algorithm performs well compared with a compact Mixed-Integer Programming model and with an Adaptive Large Neighborhood Search algorithm.

9.1 Introduction

Meetings form an important activity in modern society, both in people's business- and private life. If the occurrence of meetings is large, and the meetings are interrelated, planning them becomes a puzzle. In this article we will study the *Generalized Meeting Planning problem*.

Every day numerous secretaries have to solve the following planning problem: Find a meeting time for a number of persons to meet. If the secretary is using a standard electronic calendar system, there usually exists a function for finding the first time-interval where all persons are available. This is a simple function and very useful for the meeting planners. If the secretary has to plan more meetings, the function works as well, unless one or more of the persons have

[1]Submitted and under revision at *Computers and Operations Research (2013)*

to participate in several meetings. Then planning one meeting renders these persons unavailable to participate in other meetings in the same time-interval, if we assume that a person can only participate in one meeting at a time.

The problem of planning a number of meetings in time-intervals (time slots), some of which requires the participation of the same person, is the Generalized Meeting Planning (GMP) problem. We need to stress that the term *meeting* should be understood in its most general form: A meeting consists of a number of *entities* meeting each other in a specific time-interval. The entities can be managers, teachers and students, or any other type of resource attending the meetings.

There are naturally many variations of the GMP problem, both in the objective, i.e. what should be achieved, and in the constraints, i.e. which extra requirements are there. On the other hand, all GMPs share the same basic structure and a planning approach, i.e. optimization, for one version of a GMP may benefit from approaches for other GMPs.

The Mixed-Integer Programming (MIP) model that we will present makes it obvious to apply a Branch & Price (B&P) algorithm to find the optimal solution, due to an exponential number of variables. This works by splitting the model in a *master problem* and a number of *subproblems*. The Linear Programming (LP) relaxation of the master problem, denoted the Relaxed Master Problem (RMP), will be solved using a Column Generation (CG) algorithm. To find integer solutions by means of this LP relaxation, a Branch & Bound scheme is used, which thereby leads to a B&P algorithm. B&P algorithms have been very successful for a large number of hard optimization problems, e.g. Savelsbergh (1997); Mehrotra and Trick (2007). For an excellent introduction to B&P we refer to Lübbecke and Desrosiers (2005).

This paper is based on a technical report Bagger (2012), in which a number of different approaches to the GMP problem are attempted. We will refer to this technical report for details on various matters throughout this paper.

The outline of the paper is as follows. In Section 9.2 we will briefly survey previous approaches. In Section 9.3 we define the GMP problem as a MIP model. We will test our approach on two different variations of the GMP problem which are described in Section 9.4: The Parental Consultations Timetabling problem in Section 9.4.1, and the Supervisor Consultation Timetabling Problem in Section 9.4.2. The problems are formulated more generally as the Consultation Timetabling Problem. In both tests, we will use real-life data from a large number of Danish high schools. Computational results are presented in Section 9.5. Finally we will give a conclusion in Section 9.6.

9.2 Previous Approaches

The generalized version of the meeting planning problem considered is widely applicable, as the required structure is simple; A number of meetings between entities (e.g. individuals) are to be scheduled, and each entity can participate in multiple meetings. Hence both *entity* and *meeting* are actually used as abstracts terms here, since their definitions are problem-specific. To the best of our knowledge, the abstract form of this problem has not been considered before in the literature. However, several well-described problem domains exist for which the GMP is applicable, especially within timetabling problems in the educational sector.

In case of *High School Timetabling*, a meeting could be defined as a *lecture*, and entities could be students, teachers, classes, or a combination of these. Santos et al. (2010) describes a Cut and Column Generation algorithm for class/teacher timetabling at Brazilian high schools, where the set of teachers is used as entities. Papoutsis et al. (2003) also uses teachers and entities, and schedules for Greek high schools.

The field of *University Timetabling* is dominated by heuristic approaches. The problem is closely related to graph coloring and many early approaches were based on graph coloring heuristics, but recently meta-heuristics have gained the most interest Lewis (2008). However some researchers have applied column generation approaches. Qualizza and Serafini (2005) describes an approach where a column represents the assigning of time slots for a single course. Computational results for a single problem instance shows convincing results.

For *Examination Planning* many different approaches have been suggested, though most heuristic. A survey by Qu et. al in Qu et al. (2006) considers the recent research applied in the field. Here it is noted that decomposition methods have not attracted much attention. One of the reasons is that some soft constraints cannot be evaluated in the decomposition and so global optimality may be missed. Like for university timetabling, most of the early approaches applied graph-heuristics, but lately methods such as Tabu Search (Di Gaspero and Schaerf, 2001; Pais and Amaral, 2008) and Simulated Annealing Thompson and Dowsland (1996a,b) have gained attention. In Thompson and Dowsland (1996a) it is mentioned that this method resulted in an implementation which were used at University of Wales Swansea. Defining a single exam as a meeting between students (usually only a single student) and a group of teachers would allow applying the terminology of this paper.

In this paper we consider the case of the *Consultation Timetabling Problem* for high schools in Denmark, which essentially consists of meetings between a group of teachers and a single student. This problem is only described in the literature in Kristiansen et al. (2013).

9.3 A Mixed-Integer Programming model of the Generalized Meeting Planning problem

There are numerous ways to model the GMP problem using MIP models. In Bagger (2012) a number of these are presented, including a compact model (i.e. a model with a polynomially limited number of variables and constraints). Here we will only present what is known as the *Entity Pattern model*, as this performed best wrt. computational results.

An *entity* is a resource that has to participate in a number of meetings. Let the set of entities which are part of the GMP problem be given by the set E, indexed by $e \in E$. Let further the set of time slots be given by B, indexed by $b \in B$. Each meeting concerns a group g of entities which should meet and which is part of a set of groups G, i.e. $g \in G$.

Given the three sets E, B and G, further data about the problem is given: $\alpha_{g,b}$ is the profit of scheduling a meeting $g \in G$ to time slot $b \in B$. The incidence matrix A_g^e defines the groups such that $A_g^e = 1$ if entity $e \in E$ belongs to group $g \in G$ and $A_g^e = 0$ otherwise.

In the Entity Pattern model we will use two types of variables: The binary variable $x_{g,b}$ which defines that group $g \in G$ meets in time slot $b \in B$, and the binary variable $\lambda^{e,p}$ which defines whether the meetings of entity $e \in E$ follows the patterns $p \in P$. A pattern is essentially a schedule for an entity, which defines the time slots in which the entity is attending a meeting, but does not explicit denote which meetings. The number of patterns for an entity may be exponential. We will here assume to know the entire set of patterns P_e for each entity, and we will also assume to know the cost $\beta^{e,p}$ for using entity pattern $p \in P_e$. Let incidence matrix $M_b^{e,p}$ take value 1 if pattern $p \in P_e$ does *not* allow a meeting in time slot $b \in B$.

The entire Entity Pattern model can now be formulated, see Model 1. This constitutes the master problem in our CG formulation. The notation used is lazy; Let $\forall a$ be shorthand for $\forall a \in A$, and let $\sum_a$ be shorthand for $\sum_{a \in A}$.

Model 1 consists of one objective function and three types of constraints: The objective function given by eq. (9.3.1a) weighs the two terms, the revenue for allocating a meeting and the revenue of the entity pattern for each entity. What exactly is contained in the revenue term of the groups α and the revenue term of the patterns β is dependent on the actual meeting planning problem. In the settings we consider, different properties of the patterns are penalized, e.g. too many consecutive meetings, idle time slots and so on. It is desired to minimize these penalties, so we let β take non-positive values. The constraint in eq. (9.3.1b) ensures that each meeting can be arranged at most one time. The constraint in eq. (9.3.1c) links the group allocation variables $x_{g,b}$ with the entity patterns $\lambda^{e,p}$, i.e. ensures that an entity can only be assigned a time slot which is allowed. Finally the convexity constraint given by eq. (9.3.1d) ensures that exactly one entity pattern is chosen for each entity.

Of all the approaches attempted in Bagger (2012), Model 1 performed best in the tests and it

$$\max \quad \sum_{g,b} \alpha_{g,b} \cdot x_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p} \tag{9.3.1a}$$

$$\text{s.t.} \quad \sum_{b} x_{g,b} \leq 1 \quad \forall g \tag{9.3.1b}$$

$$\sum_{g} A_g^e \cdot x_{g,b} + \sum_{p \in P_e} \lambda^{e,p} \cdot M_b^{e,p} = 1 \quad \forall e, b \tag{9.3.1c}$$

$$\sum_{p \in P_e} \lambda^{e,p} = 1 \quad \forall e \tag{9.3.1d}$$

$$x_{g,b}, \lambda^{e,p} \in \{0,1\} \tag{9.3.1e}$$

Model 1: Master problem of the pattern formulation.

indeed possess a number of features which makes it attractive:

1. Model 1 can be converted into a set-partition model. These models are very well studied which means that the theory is well-developed and a number of good performing heuristics are known.
2. The entity patterns can be generated in the subproblem when using a CG algorithm, hence overcoming the problem of the exponential number of $\lambda^{e,p}$ variables.
3. Relations between meetings for one entity in the entity pattern is handled in the subproblem of the CG algorithm.
4. The $x_{g,b}$ variables can be used in a B&P algorithm, such that branching is performed only on these variables. A proof of this will be given in Section 9.3.2.
5. Branching only on the $x_{g,b}$ variables does not change the structure of the subproblem, as opposed to other branching schemes, facilitating a simpler B&P algorithm.

Because of the above reasons, we consider the entity pattern Model 1 a promising model. There are however a couple of weaknesses which also deserves to be mentioned:

1. Any relation between entity patterns, i.e. between the meetings of different entities becomes more cumbersome to model.
2. There is no definition of where the meetings should take place, e.g. allocation of meetings to rooms. For simplicity we have chosen to ignore this issue in this paper.

9.3.1 Obtaining Dual Bounds

When solving the RMP problem a CG algorithm will be used. Using this approach only a subset of the patterns are initially included in the model. In this case all the $x_{g,b}$ variables are initially included in the RMP and one $\lambda^{e,p}$ variable for each entity derived from an initial solution. This problem is called the Restricted Master Problem and the dual information of the solution is used to generate new variables (columns) to include in the model. When using CG to solve the RMP a dual bound can be calculated in each iteration. Let υ_g, $\pi_{e,b}$ and μ_e be the dual variables of the constraints (9.3.1b), (9.3.1c) and (9.3.1d) respectively. Consider some iteration of the CG procedure and let $(\overline{\upsilon}_g, \overline{\pi}_b^e, \overline{\mu}_e)$ be the dual solution of the Restricted Master Problem. Then the subproblem consists of finding the pattern with the highest reduced cost for each entity. For an entity e the subproblem can be denoted in the following way:

$$z_{\text{SP}}^e(\overline{\pi}, \overline{\mu}) = \max \left\{ \beta^{e,p} - \sum_{b} \overline{\pi}_b^e \cdot M_b^{e,p} - \overline{\mu}_e \;\middle|\; p \in P_e \right\} \tag{9.3.2}$$

Let $\left(x^*_{g,b}, \lambda^{e,p,*}\right)$ be the optimal solution to the RMP and let $\left(\overline{x}_{g,b}, \overline{\lambda}^{e,p}, \overline{v}_g, \overline{\pi}^e_b, \overline{\mu}_e\right)$ be the primal-dual solution to the Restricted Master Problem in any iteration. Furthermore let the optimal solution for the subproblem for entity $e \in E$ given the dual solution be denoted $\overline{z}^e_{\text{SP}}(\overline{\pi}, \overline{\mu})$. Lastly because of the strong duality theorem we have that:

$$\sum_{g,b} \alpha_{g,b} \cdot \overline{x}_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \overline{\lambda}^{e,p} = \sum_g \overline{v}_g + \sum_{e,b} \overline{\pi}^e_b + \sum_e \overline{\mu}_e \tag{9.3.3}$$

So a bound on the gap between the optimal solution to the RMP and the current considered solution in the Restricted Master Problem can be calculated:

$$\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*} - \sum_g \overline{v}_g - \sum_{e,b} \overline{\pi}^e_b - \sum_e \overline{\mu}_e \leq \tag{9.3.4}$$

$$\begin{aligned}
&\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*} \\
&\quad - \sum_{g,b} \overline{v}_g \cdot x^*_{g,b} - \sum_{e,b} \overline{\pi}^e_b \cdot \left(\sum_g A^e_g \cdot x^*_{g,b} + \sum_{p \in P_e} \lambda^{e,p,*} \cdot M^{e,p}_b \right) - \sum_{e,p \in P_e} \overline{\mu}_e \cdot \lambda^{e,p,*} =
\end{aligned} \tag{9.3.5}$$

$$\sum_{g,b} \left(\alpha_{g,b} - \overline{v}_g - \sum_e A^e_g \cdot \overline{\pi}^e_b \right) \cdot x^*_{g,b} + \sum_{e,p \in P_e} \left(\beta^{e,p} - \sum_b \overline{v}^e_b \cdot M^{e,p}_b - \overline{\mu}_e \right) \cdot \lambda^{e,p,*} \tag{9.3.6}$$

The step from eq. (9.3.4) to eq. (9.3.5) is done by using eq. (9.3.1b), (9.3.1c) and (9.3.1d) since they must hold for the optimal solution. The step from eq. (9.3.5) to eq. (9.3.6) is merely a rearrangement of the terms.

If the RMP is dualized then, for a group g and a time slot b, one of the constraints that are obtained is $v_g + \sum_e A^e_g \cdot \pi^e_b \leq \alpha_{g,b}$. Since $x_{g,b}$ are non-negative variables then this means that $\sum_{g,b} \left(\alpha_{g,b} - \overline{v}_g - \sum_e A^e_g \cdot \overline{\pi}^e_b\right) \cdot x^*_{g,b} \leq 0$ thus removing this term from (9.3.6) provides an upper bound. Now consider the coefficient of the $\lambda^{e,p,*}$-variables in (9.3.6). These correspond to the objective function of the subproblems in the current iteration of the CG algorithm. Since $\overline{z}^e_{\text{SP}}(\overline{\pi}, \overline{\mu})$ is the objective value of the optimal pattern for the subproblem of entity e then each coefficient must be bounded by this value, i.e.

$$\sum_{e,p \in P_e} \left(\beta^{e,p} - \sum_b \overline{v}^e_b \cdot M^{e,p}_b - \overline{\mu}^e \right) \cdot \lambda^{e,p,*} \leq \sum_e \overline{z}^e_{\text{SP}}(\overline{\pi}, \overline{w}) \cdot \left(\sum_{p \in P_e} \lambda^{e,p,*} \right) \tag{9.3.7}$$

As (9.3.1d) constrains $\sum_{p \in P_e} \lambda^{e,p,*}$ to be one for each entity e then we can replace this sum by one and so an upper bound on the optimal solution of the RMP can be calculated:

$$\sum_{g,b} \alpha_{g,b} \cdot x^*_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p,*} \leq \sum_{g,b} \alpha_{g,b} \cdot \overline{x}_{g,b} + \sum_{e,p \in P_e} \beta^{e,p} \cdot \overline{\lambda}^{e,p} + \sum_e \overline{z}^e_{\text{SP}}(\overline{\pi}, \overline{\mu}) \tag{9.3.8}$$

This means that if all the subproblems are solved to optimality then we can obtain a dual bound for the RMP in every iteration of the CG algorithm at a negligible computational cost. So in a B&P algorithm this upper bound can be checked against the incumbent in each iteration of the CG algorithm. If the bound gets below the incumbent in some iteration, the node can be fathomed before all the columns are generated instead of putting off the fathoming until the node is solved to optimality. The drawback of this approach is that we need to generate the optimal columns in each iteration to get the correct bound. However this can be circumvented by replacing $\overline{z}^e_{\text{SP}}$ with an upper bound on the subproblem (9.3.2). In our testing we will only add the optimal solutions of the subproblem.

9.3.2 Branching in the Branch & Price Algorithm

Model 1 has the important feature that it is possible to only branch on the $x_{g,b}$ variables. This simplifies the algorithm significantly because the standard B&P branching techniques can be avoided. In the remainder of this section we will show that branching on only the $x_{g,b}$ variables is indeed enough to obtain the optimal integer solution, i.e. that branching on the $\lambda^{e,p}$ variables is never necessary. To show this we will use the properties of *perfect matrices* Ryan and Falkner (1988).

Definition. (Padberg (1974); Ryan and Falkner (1988)) *An $m \times n$ zero-one set-partitioning polyhedron characterized by*
$\{x | x \in \mathbb{R}^n, Ax = e, x \geq 0\}$ $\left(\text{where } e = (1, 1, \ldots, 1)^T\right)$ *is said to be perfect if it has only zero-one integral vertices.*

Assume that in some node in the Branch & Bound tree the optimal solution to the RMP is integral on the $x_{g,b}$ variables and that all the branching performed until this node has only been done on the $x_{g,b}$ variables. Let $\overline{x}_{g,b}$ be the solution and assume that all feasible patterns for the entities are in the model. If the $x_{g,b}$ variables are substituted with the values $\overline{x}_{g,b}$ in Model 1 then the remaining problem will be as in Model 9.

$$\max \quad \sum_{e,p \in P_e} \beta^{e,p} \cdot \lambda^{e,p} \tag{9.3.9a}$$

$$\text{s.t.} \quad \sum_{p \in P_e} \lambda^{e,p} \cdot M_b^{e,p} = 1 - \sum_g A_g^e \cdot \overline{x}_{g,b} \quad \forall e, b \tag{9.3.9b}$$

$$\sum_{p \in P_e} \lambda^{e,p} = 1 \quad \forall e \tag{9.3.9c}$$

$$\lambda^{e,p} \in \{0, 1\} \tag{9.3.9d}$$

Model 9: Master problem of the pattern formulation given the integral solution $\overline{x}_{g,b}$.

Consider any entity e. If $\sum_g A_g^e \cdot \overline{x}_{g,b} = 1$ for any time slot b then the corresponding constraint and every pattern $p \in P_e$ where $M_b^{e,p} = 1$ can be removed from the model. Removing all the constraints and patterns fulfilling the latter creates a zero-one set partitioning problem. Note that the model is decomposable by the entities, i.e. the optimal solution for an entity is independent of the remaining entities. The problem is very easy to solve since each entity can only be assigned one pattern and each of the patterns can be deduced from the $\overline{x}_{g,b}$ values. However since the solution is based on the RMP it could be the case that fractional patterns were chosen and further branching were needed on the $\lambda^{e,p}$ variables.

To see if there exists an integral solution to the problem in Model 9, it could be checked whether the set-partitioning polyhedron is perfect. However this might not be so easy to show but, as mentioned earlier, since the problem is decomposable on the entities then subproblems can be generated by decomposing Model 9 on the entities. If the set partitioning polyhedron of each of these subproblems are perfect then the subproblems has integral optimal solutions meaning that the overall problem has an integral optimal solution. To see that the subproblems are perfect we need Definition 9.3.2 and Theorem 9.3.2.

Definition. (Padberg (1974); Ryan and Falkner (1988)) *An $m \times k$ zero-one matrix A_k with $k \leq m$ is said to have the property $\Pi_{\beta,k}$ if:*

- *A_k contains a $k \times k$ non-singular submatrix B_k where all rows and columns sum to β.*
- *Every row in A_k which is not in B_k is either equal to a row in B_k or the sum of the row is strictly less than β.*

Definition. (Padberg (1974); Ryan and Falkner (1988)) *A perfect $m \times n$ zero-one matrix A does not contain any $m \times k$ submatrix A_k where $3 \leq k \leq m$ with the property $\Pi_{\beta,k}$ where $\beta \geq 2$.*

Consider the subproblem for any entity and let A be the corresponding zero-one matrix which could look like the following:

$$A = \begin{bmatrix} 1 & 1 & & 1 & \cdots & 1 \\ & 1 & 1 & & \cdots & \\ 1 & & & 1 & \cdots & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ & 1 & & 1 & \cdots & 1 \\ 1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}$$

Note the last row in the matrix which corresponds to the convexity constraint for the entity and thus has ones in all the columns. This row ensures that the matrix does not contain a submatrix A_k with property $\Pi_{\beta,k}$.

Definition. *A is a perfect matrix.*

Proof of Claim 9.3.2. To prove that A is a perfect matrix we will show by contradiction that A fulfills Theorem 9.3.2. Assume that there is a submatrix A_k of A with property $\Pi_{\beta,k}$ and let B_k be the corresponding $k \times k$ non-singular submatrix of A_k where all rows and columns sum to β. Consider the last row r corresponding to the convexity constraint. We will give the following two claims:

Definition. *Row r cannot be one of the rows in the submatrix B_k.*

Definition. *Row r cannot be one of the remaining rows of A_k not in B_k.*

Assuming that both Claim 9.3.2 and 9.3.2 are true then clearly they present a contradiction thus Claim 9.3.2 must hold. This concludes that A is a perfect matrix. ∎

Proof of Claim 9.3.2. We know that row r sums to k in the matrix A_k since it has ones in all columns. This means that if r is in B_k then all the other rows must also sum to k which can only be done if they have ones in all columns as well. This contradicts that B_k is non-singular and so r cannot be one of the rows in B_k. ∎

Proof of Claim 9.3.2. All the rows in B_k must sum to β which must strictly be less than k, otherwise B_k would not be non-singular as mentioned in the proof of Claim 9.3.2. Since row r sums to k then this contradicts the assumption that A_k had property $\Pi_{\beta,k}$ as this would imply that the sum of row r should be strictly less than $\beta < k$. Therefore the row r cannot be one of the rows in A_k which is not in B_k. ∎

Since Claim 9.3.2 holds then if the $x_{g,b}$ variables are integral in the optimal solution for the RMP in some node of the B&P tree then there exists an integral optimal solution for the RMP in that node. The patterns that can be deduced from the $x_{g,b}$ variables are the only patterns that can be used in an integral solution because of the convexity constraints and so these patterns must be the optimal patterns to choose. This means that the branching can be done solely on the $x_{g,b}$ variables, i.e. the subproblems do not need to be changed throughout the branching tree.

9.3.3 Branching Selection

When branching on a fractional $x_{g,b}$ variable, one problem is that the one-branch creates a more restricted subproblem than the zero-branch. Therefore the B&P tree becomes very unbalanced. This is a well-known issue. A way to circumvent this could for instance be to use constraint branching as the Ryan-Foster branching method for which empirical studies has shown that this approach performs well on set partitioning problems Ryan and Foster (1981). The branching is

performed by identifying two constraints r and s. Either these two constraints must be fulfilled by the same variable or by different variables. So if a_{rj} and $a_{s,j}$ are the coefficients associated with the variable x_j in the constraints r and s respectively then the branching is done in the following way:

$$\sum_{j:a_{rj}=a_{sj}=1} x_j = 0 \quad \vee \quad \sum_{j:a_{rj}=a_{sj}=1} x_j = 1$$

The difference between the Ryan-Foster branching and variable branching in this context only occurs if two entities are in more than one group together. We cannot guarantee that this is true, and for the tests performed in Section 9.4 this is rarely the case, so the majority of the Ryan-Foster branching will result in branching on a single variable. Therefore we have decided to do a variable branching instead.

So the choice resides on how to do the variable branching. A usual approach is to branch on a variable x which is *most fractional*, i.e. a variable where the fractional part is closest to 0.5. Another idea is to consider a subset of all the fractional variables as potential candidates for branching, and then for each of these variables solve the relaxations of the two subproblems. This branching scheme is known as *strong branching* and the candidates we will consider are the k candidates which are *closest* to 0.65, as done by Røpke (Røpke, 2013). Instead of just solving the relaxations of all the subproblems induced by the candidates, a more clever approach to speed up strong branching is used, as described in Røpke (2012).

The main difference between our implementation and that of Røpke is the choice of the candidates to consider. The k candidates which are closest to 0.65 are chosen to be considered when branching. Røpke chose to set $k = 30$ in the beginning of the B&P algorithm and then at some point lower k to 15 Røpke (2013). The reason for this is that it is usually more important to do good branching decisions in the beginning of the B&P algorithm. Instead of choosing a depth of the tree or a number of processed nodes to be the breaking point of when to lower k, we have chosen to determine k based on the number of fixed variables. This is because when a variable is fixed to zero then in most cases this is a very weak restriction and so two subproblems at the same depth in the tree can have a great difference in the size of the problem in terms of non-fixed variables. One thing to note is that when one of the $x_{g,b}$ is fixed to one then a lot of other variables get fixed to zero. So when we calculate the number of fixed variables we will count every $|B| - 1$ variables which are fixed to zero as one variable fixed to one. This means that if in some node of the B&P tree $\mathcal{O}$ is the set of variables fixed to one and $\mathcal{Z}$ is the set of variables fixed to zero then the *counted* variable fixings is $|\mathcal{O}| + |\mathcal{Z}|/(|B| - 1)$. Then for every tenth *counted* variable fixing, we half the value of k beginning with 60.

9.4 Test Applications

In this section we describe two different applications of the GMP problem. Both of these applications originate from the Danish high school system, and the description of the problems given in the following has shown to be applicable to hundreds of different high schools. Sections 9.4.1 and 9.4.2 describe the two test-applications, and Section 9.4.3 presents a subproblem in context of GMP which can be used to solve both applications. In Kristiansen et al. (2013) it is shown that both test-applications are $\mathcal{NP}$-hard and since these are a special case of GMP this means that GMP is $\mathcal{NP}$-hard as well.

9.4.1 Parental Consultation Timetabling Problem

The first application is the *Parental Consultation Timetabling Problem* (PCTP). Few times a year, the Danish high schools, 9th to 12th school year, offer the possibility for school meetings where the parents together with the pupil meet selected teachers for short private meetings. The pupil and the parents will choose a number of teachers they wish to meet. At a certain date, the booking possibility is closed, and a secretary at the high school will use an optimization algorithm (see

Kristiansen et al. (2013)) to plan the meetings. At the day of the meeting, the parents and pupil will come to the school, and the different teachers will be spread out in different class-rooms at the school. The parents and the pupil are then given a schedule, for which teacher to meet in which time slot. The parental meetings are often placed in the evening of a normal workday to ease the attendance of the parents. Evening work is however more demanding for the teachers, so compact schedules becomes important. The GMP problem can be used to find the best schedules of both the parents and the teachers.

9.4.2 Supervisor Consultation Timetabling Problem

The second application is for *Supervisor Consultation Timetabling Problem* (SCTP). In the final year for a student at a Danish high school, the student is required to do a large study project. Each student selects two course subjects and thereby two teachers whom will be supervisors on the project. During the project process, the students are required to attend a meeting with their supervisors. These meetings are used to give the students some guidance for different parts for the project, such as problem definition or literature research.

The planning of the SCTP is very similar to the PCTP, however in SCTP the students have only one request for a meeting (where both supervisors attend). These meetings are normally placed in the daytime, hence it is often necessary to interrupt the lectures for a given student. However it is not allowed to assign a meeting during lunch breaks. As each student only has one meeting the SCTP essentially consists of finding the best schedules for the teachers.

9.4.3 Generalized Meeting Planning Subproblem for the Consultation Timetabling Problem

There are a number of requirements to be fulfilled for an entity pattern to be feasible for both PCTP and SCTP:

- For entity $e \in E$ the incidence matrix $C_b^e \in \{0,1\}$ defines whether a meeting can take place in time slot b. A teacher might be unavailable if he is occupied by other activities, such as teaching.
- For each entity $e \in E$ only a maximum number of meetings Q_e in a row can be accepted. This can either be a hard or soft constraint specified by $HS_e \in \{0,1\}$. If the constraint is hard ($HS_e = 1$) then a meeting cannot be scheduled after a sequence of Q_e meetings. If the constraint is soft ($HS_e = 0$) then a sequence can be longer than Q_e but at a cost of ω_e times the number of time slots that the sequence exceeds Q_e.

We introduce the set of days, D, indexed by $d \in D$. The parameter $V_{b,d} \in \{0,1\}$ denotes whether time slot b is at day d. Let υ_g, $\pi_{e,b}$ and μ_e be the dual variables associated with the constraints (9.3.1b), (9.3.1c) and (9.3.1d), respectively. Let $C_b^e \in \{0,1\}$ take value 1 if entity e is available in time slot b, and 0 otherwise. Let $m_b \in \{1 - C_b^e, 1\}$ be a variable denoting whether the entity does *not* have a meeting in time slot b. Let $f_d^{\text{first}} \in \{0,1\}$ and $f_d^{\text{last}} \in \{0,1\}$ indicate the first and last time slot the entity has a meeting in day $d \in D$, respectively.

The cost function for idle time slots are modeled as a piece-wise linear function by introducing the variable $v_{d,j} \in \{0,1\}$, where $j \in \{1,\ldots,m\}$, which takes value 1 if entity e has j idle time slots in day d, and 0 otherwise. The variable $y_{b,d} \in \mathbb{R}^+$ takes value 1 if the length of the sequence starting in time slot b on day d exceeds Q_e. Let variable $u_d \in \{0,1\}$ take value 1 if the entity has at least one meeting at day d, and 0 otherwise, and variable $v \in \mathbb{R}^+$ denotes the amount of days the entity has scheduled meetings, minus one.

Model 1 shows the entire subproblem.

Constraints (9.4.1b) and (9.4.1c) are not essentially needed to get feasible patterns, however they can help avoiding infeasible solutions. As an example say entity e_1 is in two groups and that the dual values make it attractive for entity e_1 to have meetings in time slots 1 and 6. Assigning meetings for e_1 in these two time slots would create a lot of idle time slots in the schedule and

$$\max(e, \overline{\pi}, \overline{\mu}) \; -\sum_{d,j} \gamma^e_{d,j} \cdot v_{d,j} - \sum_{b,d} \omega_e \cdot y_{b,d} - \zeta_e \cdot v - \sum_b \overline{\pi}_{e,b} \cdot m_b - \overline{\mu}_e \tag{9.4.1a}$$

$$\text{s. t.} \quad \sum_b x_{g,b} \leq A^e_g \quad \forall\, g \tag{9.4.1b}$$

$$\sum_g A^e_g \cdot x_{g,b} + m_b = 1 \quad \forall\, b \tag{9.4.1c}$$

$$\sum_g \sum_{b'=b}^{b+Q_{e'}} V_{b',d} \cdot A^{e'}_g \cdot x_{g,b'} \leq Q_{e'} \quad \forall\, HS_{e'} = 1, V_{b,d} = 1, \; b \leq |B| - Q_{e'} - 1, \; d, e' \neq e \tag{9.4.1d}$$

$$\sum_{b'=b}^{b+Q_e} V_{b',d} \cdot m_{b'} \geq 1 \quad \forall d, b \leq |B| - Q_e - 1, \; HS_e = 1, V_{b,d} = 1 \tag{9.4.1e}$$

$$f^{\text{last}}_d + b \cdot m_b \geq b \quad \forall\, b, d, V_{b,d} = 1 \tag{9.4.1f}$$

$$f^{\text{first}}_d - (|B| - b - 1) \cdot m_b \leq b \quad \forall\, b, d, V_{b,d} = 1 \tag{9.4.1g}$$

$$f^{\text{last}}_d - f^{\text{first}}_d + \left(1 + \frac{HS_e}{Q_e}\right) \cdot \sum_b V_{b,d} \cdot (1 - m_b) - \sum_j j \cdot v_{d,j} \leq -\frac{HS_e}{Q_e} \quad \forall\, d \tag{9.4.1h}$$

$$\sum_j v_{d,j} = 1 \quad \forall\, d \tag{9.4.1i}$$

$$\sum_{b'=b}^{b+Q_e} V_{b',d} \cdot m_{b'} + y_{b,d} \geq 1 \quad \forall d, b \leq |B| - Q_e - 1, \; HS_e = 0, V_{b,d} = 1 \tag{9.4.1j}$$

$$m_b + u_d \geq 1 \quad \forall\, b, d, V_{b,d} = 1 \tag{9.4.1k}$$

$$\sum_d u_d - v \leq 1 \tag{9.4.1l}$$

$$x_{g,b}, v_{d,j} \in \{0, 1\} \tag{9.4.1m}$$

$$m_b \in \{1 - C^e_b, 1\} \tag{9.4.1n}$$

$$f_{\text{first}}, f_{\text{last}}, y_{b,d}, u_d, v \in \mathbb{R}^+ \tag{9.4.1o}$$

Model 1: subproblem of the entity patterns formulation given the dual values $\{\overline{v}_{e,b}\}$ and $\{\overline{w}_e\}$.

therefore assigning meetings in time slot 2, 3, 4 and 5 (assuming these time slots are feasible for the entity) would remove the idle time slots. This is an infeasible pattern as e_1 is in two groups but scheduled for six meetings. Constraints (9.4.1b) ensure that only requested meeting are assigned and (9.4.1c) ensure that a group can only be placed at time slot b if that time slot is available for the entity. Constraints (9.4.1f), (9.4.1g) and (9.4.1h) denote the number of idle time slots the entity has at a given day. If an idle time slot is required after each sequence of meetings of size Q_e in (9.4.1h) (i.e. HS = 1) this is not counted as a idle time slot. Constraints (9.4.1i) are convexity constraints ensuring that only one of the idle time slot variables is set to one.

Constraints (9.4.1d) and (9.4.1e) are the hard sequential constraints. If $HS_e = 1$, a sequence of meetings cannot exceed Q_e. Constraints (9.4.1j) are the soft constraints of sequential meetings (i.e. $HS_e = 0$). The way the number of time slots that exceeds Q_e in some sequence of length k is computed by settings lower bounds of 1 on the first $k - Q_e$ of the $y_{b,d}$ variables. As an example let an optimal solution for the entity have scheduled meeting in the time slots 1, 3, 4, 5, 6, 7, 8 and 10 which are all connected to day 1 and let $Q_e = 4$ then the sequence 3, 4, 5, 6, 7, 8 exceeds Q_e by 2 meetings. This means that $y_{3,1}$ and $y_{4,1}$ both will get a lower bound of 1 and since they are not bounded by any other constraint they will be set to 1 in the optimal solution. This means that this sequence will be correctly penalized by 2 times ω_e in the objective function. Constraints

(9.4.1k) and (9.4.1l) sets the values for the variables u_d and v.

9.5 Computational Results

To evaluate the B&P algorithm we compare it with two other known solution approaches for both the PCTP and the SCTP. These are described in detail in Kristiansen et al. (2013) as well as the values of the different parameters of the model.

- *Adaptive Large Neighborhood Search*: This is a heuristic based on Adaptive Large Neighborhood Search, which is currently used by many high schools in Denmark to solve both the PCTP and the SCTP.
- *Gurobi 5.0.1*: This denotes the performance of Gurobi v. 5.0.1 on a standard MIP model (the formulation from Bagger (2012) where Gurobi performed best). The results reported here are not the same ones used in Kristiansen et al. (2013) as new runs were performed to obtain more information. Therefore small variations are seen in the data.

The implementation was done in C# 4.5 running on a Windows machine equipped with an Intel i7 CPU clocked at 2.80GHz and with 12GB of RAM. Gurobi 5.0.1 was used as LP solver for the master problem and as MIP solver for the subproblems (with default parameter settings). The maximum running time for Gurobi and B&P was set to 1 hour, while the ALNS ran 10 times with a time limit of 2 minutes (we report the average objective values). Thereby the comparison is 'unfair' in favor of Gurobi and B&P, which should be kept in mind throughout this section.

As a starting solution to the B&P algorithm we use a solution obtained by a single run of the ALNS algorithm. Since the ALNS algorithm is stochastic, the B&P algorithm is also stochastic, but for time reasons only a single run of the B&P algorithm was performed. It should be noted that the deviation between solutions obtained by ALNS for the same datasets have experimentally been shown to be low. Therefore the solutions obtained by the B&P algorithm are also expected to have low deviation between them. The single run of ALNS to get the initial solution is included in the reported running time for B&P.

All datasets have been obtained from the database of the commercial product Lectio, which is used by hundreds of high schools in Denmark. Thereby all datasets represent a case of a real-world optimization problem.

9.5.1 Parental Consultation Timetabling Problem

Table 9.1 shows the results obtained for the PCTP. Table 9.1 shows these results in summarized format.

The most prominent numbers drawn from these results are the gap to the best upper bound. This shows that the B&P algorithm in average is only 2.32% within optimum, which is slightly worse than the ALNS algorithm, but far better than the solutions Gurobi provides. Furthermore the B&P algorithm finds solutions within 5% from optimum for 92 instances.

Table 9.1: Summary of results for PCTP. 'Best obj' denotes the amount of instances where the algorithm provided the best objective value (including draws). 'Best UB' denotes the amount of instances where the algorithm found the best upper bound (including draws). Columns 'Gap $\leq q$ shows the amount of instances for which the respective algorithm provided a gap $\leq q$. 'Avg. Gap to best UB' is found for each algorithm by finding the best available UB for each instance, calculating the gap to the solution provided, and averaging these gaps.

	Best obj	Best UB	Gap = 0%	Gap $\leq$ 2%	Gap $\leq$ 5%	Avg. Gap to best UB
ALNS	46	-	-	-	-	2.31%
Gurobi	21	19	17	23	35	9.37%
B&P	54	94	16	54	92	2.32%

9.5.2 Supervisor Consultation Timetabling Problem

Also for the SCTP the B&P algorithm performs better than Gurobi, providing the best solution in 68 cases. Also in terms of bounds, the B&P algorithm outperforms Gurobi as it finds the best bound in 95 cases. On average, the B&P algorithm is 1.15% within optimum when comparing with the best found bounds which is slightly better than the ALNS algorithm. This is also significantly lower than the 7.13% obtained by Gurobi.

Table 9.2: Summary of results for SCTP. Columns are equivalent to those in Table 9.1.

	Best obj	Best UB	Gap = 0%	Gap ≤ 2%	Gap ≤ 5%	Avg. Gap to best UB
ALNS	37	-	-	-	-	1.26%
Gurobi	16	14	10	22	42	7.13%
B&P	68	95	23	80	97	1.15%

9.6 Conclusion

In this paper we have presented a generalization of a whole family of planning problems, the Generalized Meeting Planning problem. A Branch & Price algorithm has been presented which were tested on two different versions of the Generalized Meeting Planning problem: The Parental Consultation Timetabling Problem and the Supervisor Consultation Timetabling Problem. For both problems the developed B&P algorithm is tested on 100 real-world data examples from Danish high schools. The B&P algorithm on average obtains a gap of 2.32% for the Parental Consultation Timetabling Problem and 1.15% for the Supervisor Consultation Timetabling Problem. These are convincing results for effectiveness of the algorithm. We find it likely that there are many other problems where the Generalized Meeting Planning problem is applicable, and the described B&P approach therefore can be applied.

Table 9.1: Computational results for 100 datasets for the PCTP problem. Column 'ALNS' denotes the solution obtained by a problem-specific *Adaptive Large Neighborhood Search* heuristic. For Gurobi the obtained lower- and upper bound (columns 'Obj' and 'UB' respectively) is shown, as well as the final gap. The amount of required seconds is also shown. For the B&P algorithm, the final number of explored nodes in the B&P tree is shown, as well as the obtained bounds, the final gap, and total run time. Note that a node is only counted as being explored after a branching has been performed, i.e. if the optimal solution is found in the root node the number of explored nodes is 0. The best found solution is marked in **bold**, and the best found bound is marked with a '*'.

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Alleroed	51	12	38	484.8	**485.0**	*485.0	0.0	3	67	**485.0**	*485.0	0.0	167
Alssund	84	18	55	849.4	**850.6**	*850.6	0.0	503	112	850.4	850.8	0.1	>3600
Aurehoej1	537	18	189	3655.9	3429.4	3773.0	10.0	>3600	148	**3660.7**	*3763.0	2.8	>3600
Aurehoej2	409	18	170	**3219.5**	2976.6	3297.1	10.8	>3600	154	3219.3	*3272.0	1.6	>3600
Broenderslev	241	24	111	1905.6	1650.4	1965.2	19.1	>3600	88	**1910.9**	*1936.0	1.3	>3600
CPHWEST	133	39	59	**1044.5**	1001.8	1124.0	12.2	>3600	32	1041.4	*1067.8	2.5	>3600
DetKristne	247	32	83	**1830.9**	1455.6	1973.1	35.6	>3600	40	1829.5	*1900.2	3.9	>3600
Dronninglund1	108	30	36	782.0	744.0	801.5	7.7	>3600	210	**783.9**	*798.0	1.8	>3600
Dronninglund2	94	30	34	**664.5**	641.4	672.5	4.9	>3600	810	658.2	*671.8	2.1	>3600
Egaa	265	24	109	**2318.5**	2132.1	2374.0	11.4	>3600	140	2317.7	*2365.0	2.0	>3600
Egedal	408	27	186	3558.2	3091.5	3625.2	17.3	>3600	16	**3568.3**	*3608.4	1.1	>3600
Esbjerg1	345	24	124	2402.1	2333.5	2466.1	5.7	>3600	228	**2403.5**	*2440.3	1.5	>3600
Esbjerg2	307	24	160	2314.3	2119.4	2440.3	15.1	>3600	92	**2315.6**	*2365.7	2.2	>3600
Esbjerg3	255	24	94	**1839.5**	1801.8	1885.6	4.7	>3600	32	1837.8	*1860.1	1.2	>3600
Esbjerg4	351	24	126	2612.3	2451.6	2706.2	10.4	>3600	228	**2616.7**	*2657.2	1.5	>3600
Frederikssund	49	24	26	403.8	**404.2**	406.6	0.6	>3600	161	**404.2**	*404.2	0.0	1513
Frederiksvaerk	74	8	54	697.9	**699.0**	*699.0	0.0	22	168	**699.0**	*699.0	0.0	186
Gefion	479	18	220	**3958.1**	3309.4	4248.8	28.4	>3600	68	3920.6	*4175.6	6.5	>3600
Gladsaxe	901	40	302	**6950.7**	5443.7	7163.9	31.6	>3600	0	6944.2	*7142.7	2.9	>3600
Greve	336	18	152	**2482.6**	2192.9	2535.9	15.6	>3600	78	2474.3	*2532.6	2.4	>3600
Haslev1	123	18	65	1060.2	1058.0	1072.6	1.4	>3600	115	**1061.3**	*1061.3	0.0	475
Haslev2	122	18	67	988.7	977.2	1020.9	4.5	>3600	1512	**989.8**	*1004.4	1.5	>3600
Herlufsholm1	143	24	43	894.5	842.2	919.1	9.1	>3600	533	**895.1**	*905.1	1.1	>3600
Herlufsholm2	88	24	31	621.8	594.8	664.4	11.7	>3600	720	**623.6**	*633.1	1.5	>3600
Herning1	118	27	64	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Herning2	75	27	34	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Herning3	140	27	69	**2.0**	**2.0**	*2.0	0.0	0	0	**2.0**	*2.0	0.0	120
Himmelev	453	34	177	3471.7	3093.4	3735.0	20.7	>3600	10	**3483.9**	*3586.7	2.9	>3600
Hjoerring	179	30	132	**1009.2**	790.1	1231.4	55.9	>3600	0	999.2	*1178.3	17.9	>3600
HorsensGym	123	18	75	1129.4	**1132.4**	*1132.5	0.0	2544	467	1129.3	1133.8	0.4	>3600
HorsensStats	143	21	98	1253.8	1217.8	1286.3	5.6	>3600	802	**1254.2**	*1268.1	1.1	>3600
Ikast-Brande	52	30	24	**449.7**	448.5	458.6	2.3	>3600	169	**449.7**	*452.7	0.7	>3600
Johannesskolen1	165	24	58	**1188.3**	1136.3	1293.1	13.8	>3600	496	1187.1	*1212.3	2.1	>3600
Johannesskolen2	97	24	39	743.9	704.1	786.4	11.7	>3600	788	**748.0**	*761.7	1.8	>3600
Johannesskolen3	135	28	56	519.9	500.3	588.4	17.6	>3600	508	**522.0**	*548.2	5.0	>3600
Kalundborg	299	27	164	**2458.3**	2212.2	2592.0	17.2	>3600	66	2457.0	*2517.8	2.5	>3600
Kolding	80	18	43	721.0	**721.4**	723.2	0.2	>3600	998	721.1	*721.8	0.1	>3600
Langkær1	52	18	35	470.6	**471.1**	*471.1	0.0	309	269	**471.1**	*471.1	0.0	750
Langkær2	90	18	50	805.9	788.6	814.1	3.2	>3600	945	**806.0**	*810.0	0.5	>3600
Middelfart	223	27	109	**1788.1**	1653.9	1916.6	15.9	>3600	98	1784.6	*1821.5	2.1	>3600
Morsoe1	105	27	37	**804.2**	755.1	834.1	10.5	>3600	352	802.3	*824.2	2.7	>3600

Continued on next page

Table 9.1 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Morsoe2	113	27	35	**778.4**	715.1	815.0	14.0	>3600	300	773.0	*806.5	4.3	>3600
Munkensdam1	256	21	127	2198.6	1851.6	2252.1	21.6	>3600	74	**2210.4**	*2234.8	1.1	>3600
Munkensdam2	345	21	157	**2846.8**	2464.9	2930.1	18.9	>3600	42	2831.1	*2905.2	2.6	>3600
NielsSteensens1	117	36	35	757.2	704.3	781.8	11.0	>3600	38	**757.4**	*776.9	2.6	>3600
NielsSteensens2	328	30	63	**1656.6**	1414.6	1763.7	24.7	>3600	52	1655.3	*1728.0	4.4	>3600
NielsSteensens3	365	30	64	1800.3	1587.7	1895.6	19.4	>3600	50	**1800.7**	*1859.4	3.3	>3600
NielsSteensens4	234	30	52	1144.4	1086.9	1229.4	13.1	>3600	4	**1146.3**	*1191.0	3.9	>3600
NielsSteensens5	263	30	57	1557.1	1451.1	1632.9	12.5	>3600	110	**1560.5**	*1606.9	3.0	>3600
Noerre	422	18	209	3944.5	3469.9	4033.6	16.3	>3600	100	**3952.3**	*4011.7	1.5	>3600
Nordfyns	192	23	102	**1795.5**	1782.3	1858.8	4.3	>3600	412	1793.4	*1836.9	2.4	>3600
Nordsjaellands1	1187	34	232	**6597.3**	5867.3	7020.9	19.7	>3600	6	6565.1	*6971.6	6.2	>3600
Nordsjaellands2	1038	34	203	2453.2	2300.1	2624.8	14.1	>3600	50	**2460.0**	*2582.7	5.0	>3600
Nordsjaellands3	457	34	106	**2634.7**	2092.3	2858.0	36.6	>3600	8	2633.4	*2759.4	4.8	>3600
Nordsjaellands4	163	34	50	1172.4	1117.5	1209.8	8.3	>3600	92	**1173.0**	*1206.1	2.8	>3600
Nordsjaellands5	712	40	164	4460.5	3808.0	4801.9	26.1	>3600	4	**4467.5**	*4692.6	5.0	>3600
Nordsjaellands6	780	34	170	**4612.2**	4025.4	4897.2	21.7	>3600	16	4599.6	*4829.8	5.0	>3600
Nordsjaellands7	880	34	190	**2894.5**	2527.6	3048.3	20.6	>3600	22	2884.9	*3010.3	4.3	>3600
Nordsjaellands8	23	34	21	241.9	**242.1**	*242.1	0.0	6	21	**242.1**	*242.1	0.0	261
Nordsjaellands9	949	34	196	5037.1	4479.7	5518.7	23.2	>3600	12	**5056.7**	*5426.4	7.3	>3600
Nordsjaellands10	31	34	23	269.1	270.4	276.1	2.1	>3600	58	**272.2**	*272.2	0.0	479
Nyborg	119	24	58	**55.4**	**55.4**	*55.4	0.0	0	0	**55.4**	*55.4	0.0	120
Nykoebing	182	24	118	1483.1	1471.3	1495.2	1.6	>3600	661	**1484.0**	*1489.3	0.4	>3600
NZahles1	324	25	109	**2365.7**	1936.3	2456.2	26.9	>3600	44	2356.3	*2447.8	3.9	>3600
NZahles2	301	24	107	**2217.8**	2000.0	2280.1	14.0	>3600	68	2211.6	*2275.6	2.9	>3600
Odder	95	18	37	**762.7**	751.8	773.2	2.8	>3600	832	761.8	*772.6	1.4	>3600
Odsherreds	193	21	96	1595.4	1548.3	1619.7	4.6	>3600	24	**1597.5**	*1610.0	0.8	>3600
Risskov1	65	15	35	536.7	**539.7**	*539.7	0.0	577	212	**539.7**	*539.7	0.0	387
Risskov2	149	15	69	1256.9	**1264.6**	1272.0	0.6	>3600	1624	1248.6	*1270.2	1.7	>3600
Risskov3	181	15	78	1389.7	**1402.7**	*1406.5	0.3	>3600	1920	1391.1	1407.3	1.2	>3600
Roedkilde	266	18	103	**2325.2**	2218.2	2352.9	6.1	>3600	32	2321.1	*2352.7	1.4	>3600
Roedovre	779	51	291	1661.7	1482.8	2008.2	35.4	>3600	14	**1683.7**	*1812.1	7.6	>3600
Rosborg1	218	24	85	**1827.9**	1752.1	1876.2	7.1	>3600	522	1798.5	*1866.6	3.8	>3600
Rosborg2	268	28	95	2223.3	1972.9	2297.3	16.4	>3600	154	**2229.4**	*2273.5	2.0	>3600
Rosborg3	487	28	307	4750.0	4602.1	4938.2	7.3	>3600	124	**4752.9**	*4824.9	1.5	>3600
Rosborg4	235	26	94	1960.4	1643.4	2033.0	23.7	>3600	184	**1967.7**	*2015.7	2.4	>3600
Roskilde	263	48	113	**2112.8**	1663.0	2252.6	35.5	>3600	0	2107.7	*2171.1	3.0	>3600
Rybners	267	24	126	**2402.5**	1952.4	2473.4	26.7	>3600	34	2395.1	*2446.5	2.1	>3600
SanktAnnae	320	21	124	**2381.6**	2114.8	2499.1	18.2	>3600	48	2375.6	*2458.5	3.5	>3600
Skive	220	36	78	**1850.3**	1604.5	1903.2	18.6	>3600	34	1849.0	*1877.5	1.5	>3600
Slagelse	85	30	50	802.6	**805.3**	*805.3	0.0	337	164	802.3	805.4	0.4	>3600
Solroed1	341	16	148	**2436.8**	2397.5	2468.9	3.0	>3600	378	2426.7	*2468.3	1.7	>3600
Solroed2	415	16	182	**3263.0**	3140.2	3318.4	5.7	>3600	238	3256.1	*3315.8	1.8	>3600
Soroe1	369	24	145	2947.3	2648.0	3106.8	17.3	>3600	58	**2957.9**	*3042.5	2.9	>3600
Soroe2	335	33	139	**2255.0**	1615.9	2641.1	63.5	>3600	10	2247.6	*2441.3	8.6	>3600
Stenhus	221	18	121	**0.0**	**0.0**	*0.0	0.0	0	0	**0.0**	*0.0	0.0	120
Stoevring	62	24	31	520.5	**521.4**	*521.4	0.0	21	76	**521.4**	*521.4	0.0	335
Struer1	237	30	127	**1656.9**	1610.0	1794.2	11.4	>3600	160	1656.4	*1697.2	2.5	>3600
Struer2	333	30	138	**2534.0**	2094.3	2793.9	33.4	>3600	104	2532.5	*2637.4	4.1	>3600

Continued on next page

Table 9.1 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Svendborg1	96	18	79	991.1	**991.4**	*991.4	0.0	35	1619	**991.4**	*991.4	0.0	>3600
Svendborg2	134	18	82	1288.0	**1289.1**	*1289.2	0.0	142	764	1288.2	1289.4	0.1	>3600
Taarnby	791	36	244	5609.2	4587.1	*5918.9	29.0	>3600	0	**5622.2**	5927.8	5.4	>3600
UCH	104	32	110	**922.6**	**922.6**	†*922.6	0.0	4	0	**922.6**	*922.6	0.0	143
ViborgGym1	206	30	88	1434.0	1348.6	1482.5	9.9	>3600	122	**1437.9**	*1467.2	2.0	>3600
ViborgGym2	149	30	62	1133.4	1101.2	1146.9	4.2	>3600	384	**1134.4**	*1138.1	0.3	>3600
ViborgGym3	294	30	143	**2211.7**	2081.2	2275.6	9.3	>3600	118	2210.0	*2236.0	1.2	>3600
ViborgHandel	324	30	95	2526.7	2160.1	2615.9	21.1	>3600	58	**2537.4**	*2614.2	3.0	>3600
ViborgKatedral	337	40	101	**2313.4**	1755.6	2516.6	43.4	>3600	8	2304.7	*2465.0	7.0	>3600
Vordingborg1	315	16	132	**2304.3**	2201.4	2353.5	6.9	>3600	38	2304.2	*2349.2	2.0	>3600
Vordingborg2	239	16	115	1924.6	1893.2	1949.4	3.0	>3600	436	**1926.9**	*1949.0	1.1	>3600

† A wrong upper bound was reported in Kristiansen et al. (2013).

Table 9.2: Computational results for 100 datasets for the SCTP. Columns have same meaning as in Table 9.1.

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Aabenraa	226	60	279	**2387.5**	2111.1	†2490.8	18.0	>3600	0	2384.9	*2465.1	3.4	>3600
Broendby1	69	21	87	**683.2**	672.1	704.9	4.9	>3600	825	682.2	*689.7	1.1	>3600
Broendby2	69	14	85	768.6	770.9	779.4	1.1	>3600	2401	**772.0**	*772.4	0.0	>3600
Broendby3	62	24	80	614.8	603.9	633.7	4.9	>3600	669	**617.7**	*620.7	0.5	>3600
Broenderslev1	115	69	148	**1302.4**	1276.0	†*1341.7	5.2	>3600	0	1300.7	1347.2	3.6	>3600
Broenderslev2	115	102	148	**1236.0**	1142.4	†*1271.8	11.3	>3600	0	1234.9	1443.5	16.9	>3600
Christianshavns	210	43	254	**2223.7**	1723.3	†2386.1	38.5	>3600	44	2217.2	*2264.6	2.1	>3600
Dronninglund1	134	100	166	1453.5	1382.4	†*1480.7	7.1	>3600	0	**1454.7**	1668.1	14.7	>3600
Dronninglund2	134	60	165	1537.5	1530.2	†*1561.7	2.1	>3600	0	**1539.8**	1568.8	1.9	>3600
Egaa	214	29	259	2376.2	2228.1	2458.3	10.3	>3600	88	**2380.4**	*2408.5	1.2	>3600
Falkoner1	64	30	83	668.0	**671.6**	677.3	0.9	>3600	36	**671.6**	*671.6	0.0	216
Falkoner2	206	37	255	**2266.1**	2056.3	†2345.0	14.0	>3600	46	2265.2	*2297.4	1.4	>3600
Falkoner3	64	30	83	**664.9**	**664.9**	670.4	0.8	>3600	178	**664.9**	*664.9	0.0	554
Grenaa1	122	28	154	1325.4	1291.8	1379.6	6.8	>3600	164	**1328.5**	*1348.1	1.5	>3600
Grenaa2	122	24	154	1290.3	1264.9	1329.6	5.1	>3600	361	**1295.9**	*1306.7	0.8	>3600
Greve1	157	28	205	1693.3	1620.6	1761.3	8.7	>3600	428	**1695.9**	*1710.0	0.8	>3600
Greve2	259	62	322	2913.5	2627.9	†3035.3	15.5	>3600	0	**2915.1**	*3025.2	3.8	>3600
Greve3	51	20	67	566.3	**566.4**	570.7	0.8	>3600	8	**566.4**	*566.4	0.0	138
Gribskov1	182	74	226	1787.6	1269.7	†1912.1	50.6	>3600	0	**1788.9**	*1849.1	3.4	>3600
Herlev1	71	24	96	730.2	728.3	750.7	3.1	>3600	980	**730.7**	*734.0	0.4	>3600
Herlev2	78	29	106	750.2	729.5	792.7	8.7	>3600	442	**750.7**	*764.8	1.9	>3600
Hoeng1	66	21	85	621.5	605.6	688.0	13.6	>3600	4	**621.8**	*621.8	0.0	139
Hoeng2	98	23	123	1038.5	1036.0	1070.2	3.3	>3600	84	**1041.4**	*1041.4	0.0	298
Hoeng3	45	22	62	408.3	387.0	479.9	24.0	>3600	20	**409.8**	*409.8	0.0	167
Hoeng4	56	23	78	589.2	590.0	612.4	3.8	>3600	368	**591.9**	*591.9	0.0	721
Koebenhavns1	143	16	180	1242.0	**1246.7**	1273.4	2.1	>3600	1406	1243.5	*1256.4	1.0	>3600
Koebenhavns2	100	16	127	785.6	**786.4**	*786.5	0.0	197	2852	784.9	786.9	0.2	>3600
Koebenhavns3	100	16	127	**725.8**	**725.8**	*725.8	0.0	23	2672	**725.8**	*725.8	0.0	2991
Koebenhavns4	146	25	184	1424.6	1406.0	1486.1	5.7	>3600	842	**1424.7**	*1432.5	0.5	>3600
Koege1	255	30	285	2333.2	1039.0	2474.3	138.1	>3600	416	**2357.1**	*2388.2	1.3	>3600
Koege2	261	36	303	2045.7	1790.3	†2278.1	27.2	>3600	304	**2056.8**	*2075.2	0.9	>3600
Koege3	258	74	288	**2622.9**	2381.7	†2890.0	21.3	>3600	0	2621.3	*2809.1	7.2	>3600
Kolding1	219	24	263	2348.7	2287.1	2422.6	5.9	>3600	666	**2354.3**	*2364.0	0.4	>3600
Kolding2	174	45	220	1908.8	1727.2	†2003.2	16.0	>3600	28	**1910.9**	*1936.4	1.3	>3600
Langkaer1	215	62	255	**2239.4**	1661.7	†2465.4	48.4	>3600	0	2230.2	*2307.5	3.5	>3600
Langkaer2	216	60	256	**2240.1**	1746.1	†2477.1	41.9	>3600	0	2239.6	*2303.6	2.9	>3600
Langkaer3	216	60	256	2258.3	1950.4	†2470.4	26.7	>3600	0	**2267.4**	*2303.7	1.6	>3600
Langkaer4	57	30	72	566.2	562.5	594.5	5.7	>3600	398	**573.4**	*573.4	0.0	1128
Langkaer5	217	56	256	**2253.3**	1982.9	†2494.2	25.8	>3600	6	2252.3	*2299.1	2.1	>3600
Langkaer6	56	62	71	**629.2**	623.3	†652.0	4.6	>3600	128	629.1	*632.1	0.5	>3600
Mariagerfjord1	123	29	154	**1318.6**	1216.6	1385.9	13.9	>3600	266	1315.9	*1331.2	1.2	>3600
Mariagerfjord2	123	29	154	1345.3	1282.9	1401.3	9.2	>3600	172	**1347.6**	*1362.7	1.1	>3600
Marselisborg1	102	22	132	**1045.4**	1019.3	1090.3	7.0	>3600	1130	1044.5	*1053.4	0.8	>3600
Marselisborg2	106	17	138	1035.4	1035.4	1047.3	1.2	>3600	694	**1037.1**	*1037.1	0.0	995

Continued on next page

Table 9.2 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Marselisborg3	105	22	132	1098.3	1076.3	1155.4	7.4	>3600	126	**1106.0**	*1106.0	0.0	524
Marselisborg4	96	17	126	947.2	**948.0**	952.9	0.5	>3600	327	947.7	*951.2	0.4	>3600
Munkensdam	191	43	225	2067.9	1741.7	†2201.2	26.4	>3600	14	**2075.6**	*2111.2	1.7	>3600
Noerresundby	303	31	367	3291.7	3131.4	3460.7	10.5	>3600	132	**3304.5**	*3320.1	0.5	>3600
Nordfyns1	173	22	207	1926.6	1884.0	1972.4	4.7	>3600	398	**1930.3**	*1940.3	0.5	>3600
Nordfyns2	173	21	206	**1929.9**	1905.6	1975.0	3.6	>3600	488	1929.4	*1945.9	0.9	>3600
Nordfyns3	173	22	207	1908.1	1870.5	1972.9	5.5	>3600	464	**1911.8**	*1924.5	0.7	>3600
Nordfyns4	173	21	215	1478.3	1452.7	1536.9	5.8	>3600	486	**1478.6**	*1499.8	1.4	>3600
NZahles1	69	13	90	619.1	615.6	634.8	3.1	>3600	4625	**620.1**	*620.5	0.1	>3600
NZahles2	62	13	78	509.5	511.7	522.4	2.1	>3600	5453	**513.0**	*513.1	0.0	>3600
Odsherreds	119	49	150	1289.4	1211.3	†1367.2	12.9	>3600	24	**1291.0**	*1305.7	1.1	>3600
Oeregaard1	219	20	258	2258.1	2257.3	2295.7	1.7	>3600	492	**2259.3**	*2289.9	1.4	>3600
Oeregaard2	213	20	256	1743.9	1749.4	1810.2	3.5	>3600	362	**1774.0**	*1798.0	1.4	>3600
Oeregaard3	219	20	258	2340.2	2311.5	2372.4	2.6	>3600	338	**2343.2**	*2361.5	0.8	>3600
Oeregaard4	219	20	258	**2339.9**	2325.0	2371.7	2.0	>3600	378	2334.1	*2363.1	1.2	>3600
Risskov	215	36	250	**2353.2**	2165.8	†2426.8	12.1	>3600	26	2352.6	*2389.7	1.6	>3600
Roedkilde	230	18	282	**2495.7**	2473.3	2534.6	2.5	>3600	790	**2495.7**	*2523.0	1.1	>3600
Rosborg1	257	22	310	2837.4	2787.5	2895.3	3.9	>3600	182	**2839.7**	*2893.3	1.9	>3600
Rosborg2	257	22	311	2805.4	2640.8	2859.4	8.3	>3600	182	**2805.5**	*2838.8	1.2	>3600
SanktAnnae1	149	23	191	1580.4	1539.7	1671.3	8.6	>3600	462	**1584.5**	*1599.6	1.0	>3600
SanktAnnae2	165	24	209	1753.4	1689.4	1844.9	9.2	>3600	250	**1754.5**	*1773.3	1.1	>3600
SanktAnnae3	21	17	29	**197.9**	**197.9**	*197.9	0.0	7	45	**197.9**	*197.9	0.0	142
SanktAnnae4	162	31	201	**1598.9**	1415.3	1718.0	21.4	>3600	108	1593.4	*1634.7	2.6	>3600
Skanderborg1	232	57	291	**2547.4**	2131.4	†2640.6	23.9	>3600	0	2545.4	*2588.6	1.7	>3600
Skanderborg2	229	60	276	2320.4	2000.8	†2414.4	20.7	>3600	0	**2324.7**	*2374.6	2.2	>3600
Skive1	140	16	182	1430.9	1420.9	1459.2	2.7	>3600	1088	**1436.1**	*1444.9	0.6	>3600
Skive2	103	31	143	995.5	856.3	1061.3	24.0	>3600	126	**996.5**	*1025.7	2.9	>3600
Skive3	140	31	182	1372.7	1307.0	1451.9	11.1	>3600	144	**1373.1**	*1401.6	2.1	>3600
Skive4	21	16	31	227.6	**227.8**	*227.8	0.0	12	24	**227.8**	*227.8	0.0	141
Skive5	98	16	131	960.7	**963.6**	971.7	0.8	>3600	234	959.1	*965.4	0.7	>3600
Skive6	110	16	145	**1119.3**	1111.6	1143.1	2.8	>3600	298	1115.3	*1131.1	1.4	>3600
Skive7	134	31	179	1284.6	1169.3	1365.8	16.8	>3600	114	**1289.7**	*1310.0	1.6	>3600
Skive8	107	16	143	**1007.1**	1005.7	1016.5	1.1	>3600	2423	1006.9	*1008.9	0.2	>3600
Skive9	100	31	133	**983.0**	959.0	1034.9	7.9	>3600	358	979.1	*1000.7	2.2	>3600
Soenderborg1	234	22	298	**2475.3**	2262.0	2590.4	14.5	>3600	104	2456.3	*2515.5	2.4	>3600
Soenderborg2	236	22	305	**2577.9**	2297.0	2701.0	17.6	>3600	118	2569.9	*2617.3	1.8	>3600
Soenderborg3	236	21	305	2597.3	2288.3	2686.6	17.4	>3600	222	**2601.5**	*2619.2	0.7	>3600
Soenderborg4	235	22	304	2554.2	2341.7	2679.2	14.4	>3600	102	**2563.3**	*2597.2	1.3	>3600
Solroed1	242	18	295	2130.3	1983.7	2180.6	9.9	>3600	646	**2132.7**	*2164.6	1.5	>3600
Solroed2	22	20	34	**228.6**	**228.6**	*228.6	0.0	0	0	**228.6**	*228.6	0.0	120
Solroed3	22	17	34	**223.3**	**223.3**	*223.3	0.0	0	0	**223.3**	*223.3	0.0	120
Solroed4	243	54	296	2354.2	736.6	†2565.5	248.3	>3600	2	**2358.2**	*2401.2	1.8	>3600
Solroed5	243	20	297	**2054.9**	1843.1	2187.6	18.7	>3600	578	2052.8	*2096.6	2.1	>3600
Solroed6	215	17	266	**1775.5**	1694.5	1833.5	8.2	>3600	846	1774.5	*1802.7	1.6	>3600
Solroed7	194	17	242	**1679.2**	1534.9	1744.7	13.7	>3600	1120	1676.5	*1703.5	1.6	>3600
Vejen1	41	10	58	**424.2**	**424.2**	*424.2	0.0	1	105	**424.2**	*424.2	0.0	137
Vejen2	126	19	159	**1198.5**	1184.2	1225.0	3.5	>3600	1182	1197.1	*1224.6	2.3	>3600

Continued on next page

Table 9.2 – Continued from previous page

					Gurobi 5.0.1				B&P				
Dataset	$\|G\|$	$\|B\|$	$\|E\|$	ALNS	Obj	UB	Gap	Time	Nodes	Obj	UB	Gap	Time
Vejen3	125	19	155	1204.6	1186.9	1234.8	4.0	>3600	1110	**1207.4**	*1214.3	0.6	>3600
Vejen4	125	19	155	1172.5	1145.2	1206.0	5.3	>3600	285	**1182.5**	*1182.5	0.0	852
Viborg1	105	19	133	1034.2	1011.4	1099.6	8.7	>3600	403	**1034.9**	*1034.9	0.0	1518
Viborg2	187	49	230	**2060.0**	1778.8	†2152.4	21.0	>3600	4	2057.8	*2108.2	2.4	>3600
Viby1	124	20	158	1256.9	1255.0	†1279.1	1.9	>3600	582	**1260.5**	*1266.3	0.5	>3600
Viby2	93	13	111	**957.5**	**957.5**	*957.5	0.0	2	0	**957.5**	*957.5	0.0	122
Viby3	45	8	62	**480.0**	**480.0**	*480.0	0.0	1	14	**480.0**	*480.0	0.0	131
Viby4	93	16	111	1053.3	**1053.5**	†*1053.5	0.0	3	2	**1053.5**	*1053.5	0.0	125
Viby5	123	21	156	1356.1	1355.6	†1374.3	1.4	>3600	554	**1356.7**	*1364.6	0.6	>3600

† A wrong upper bound was reported in Kristiansen et al. (2013).

Bibliography

N.-C. F. Bagger. Generalized Meeting Planning using Mathematical Programming. Technical report, DTU-Management, 2012.

L. Di Gaspero and A. Schaerf. Tabu search techniques for examination timetabling. In *Practice and Theory of Automated Timetabling III*, pages 104–117. Springer, 2001.

S. Kristiansen, M. Sørensen, M. B. Herold, and T. R. Stidsen. The consultation timetabling problem at danish high schools. *Journal of Heuristics*, 19(3):465–495, 2013. doi: 10.1007/s10732-013-9219-9. URL `http://dx.doi.org/10.1007/s10732-013-9219-9`.

R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR spectrum*, 30(1):167–190, 2008.

M. E. Lübbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53 (6):1007–1023, 2005. ISSN 0030364x, 15265463.

A. Mehrotra and M. A. Trick. A branch-and-price approach for graph multi-coloring. *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*, pages 15–29, 2007.

M. W. Padberg. Perfect zero–one matrices. *Mathematical Programming*, 6(2):180–196, 1974. ISSN 00255610, 14364646.

T. C. Pais and P. Amaral. Managing the tabu list length using a fuzzy inference system: an application to exams timetabling. In *The 7th International Conference for the Practice and Theory of Automated Timetabling*, pages 1–6, 2008.

K. Papoutsis, C. Valouxis, and E. Housos. A column generation approach for the timetabling problem of greek high schools. *The Journal of the Operational Research Society*, 54(3):230–238, 2003.

R. Qu, E. Burke, B. McCollum, L. T. Merlot, and S. Y. Lee. A survey of search methodologies and automated approaches for examination timetabling. *Computer Science Technical Report No. NOTTCS-TR-2006-4, UK*, 2006.

A. Qualizza and P. Serafini. A column generation scheme for faculty timetabling. In E. Burke and M. Trick, editors, *Practice and Theory of Automated Timetabling V*, volume 3616 of *Lecture Notes in Computer Science*, pages 161–173. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-30705-1.

S. Røpke. An era in vehicle routing research is coming to an end: the full solomon test set is solved to optimality, October 2012. Presentation in Recent Research Results in Operations Research at Department of Manegement Science at the Technical University of Denmark.

S. Røpke. Private communication, April 2013.

D. Ryan and J. Falkner. On the integer properties of scheduling set partitioning models. *European Journal of Operational Research*, pages 442–456, 1988.

D. Ryan and B. A. Foster. Integer programming approach to scheduling. *Computer Scheduling of Public Transport, Urban Passenger Vehicle and Crew Scheduling: Papers Based on Presentations at the International Workshop.*, pages 269–280, 1981.

H. Santos, E. Uchoa, L. Ochi, and N. Maculan. Strong bounds with cut and column generation for class-teacher timetabling. *Annals of Operations Research*, pages 1–14, 2010. ISSN 0254-5330. 10.1007/s10479-010-0709-y.

M. Savelsbergh. A branch-and-price algorithm for the generalized assignment problem. *Operations Research*, 1997.

J. Thompson and K. A. Dowsland. General cooling schedules for a simulated annealing based timetabling system. In *Practice and Theory of Automated Timetabling*, pages 345–363. Springer, 1996a.

J. M. Thompson and K. A. Dowsland. Variants of simulated annealing for the examination timetabling problem. *Annals of Operations research*, 63(1):105–128, 1996b.

Part V

Conclusion

Chapter 10

Conclusion

This thesis has highlighted some of the educational planning problems the high schools in Denmark are struggling with. Three different research subjects have been followed in this thesis; *High School Timetabling*, *Student Sectioning* and *Meeting Planning Problems*.

This chapter highlights the main contribution of the thesis by summarizing the conclusions of the works herein, followed by some general thoughts on the current trends and future research subjects within educational planning problems.
As aforementioned, the contributions of this thesis are two-fold; scientific contribution and practical contribution.

10.1 Scientific Contribution

In the first part of this thesis a comprehensive survey on educational planning problems was presented. The survey presents the four main planning problems and the research conducted on these; University Course Timetabling, High School Timetabling, Examination Timetabling and Student Sectioning. This is the first survey on Student Sectioning.

The remaining of the thesis has been partitioned into three research directions;

High School Timetabling: This thesis has contributed with two scientific papers concerned High School Timetabling on the XHSTT format.

An efficient Adaptive Large Neighborhood Search (ALNS) has been applied and submitted as a contribution to the Third International Timetabling Competition in 2011 (ITC2011). The algorithm was among the finalists and finished third.

Furthermore, an exact solution method for the XHSTT has been developed. A Mixed Integer Programming (MIP) model has been created and made it possible to establish non-trivial lower bounds on several instances and prove optimality of a few.

Student Sectioning: Three research papers have been conducted on Student Sectioning problems. The papers are concentrated on two different problems at the high schools in Denmark; Elective Course Student Sectioning (ECSS) and High School Student Sectioning (HSSS).

ECSS is the problem of assigning 2nd and 3rd year students to elective course classes, based on their requests. Two solution methods have been applied. The first method is based on Dantzig-Wolfe decomposition in a Branch-and-Bound framework. This method proves that a previously applied method is insufficient and the model is lacking some restrictions. A more comprehensive MIP model has then been created for the ECSS and solved using an ALNS algorithm. This model includes fairness distribution of the students and it is possible to provide solutions with an average gap of 1% from optimum.

The second Student Sectioning problem at the Danish high schools is the problem of assigning first year students to, 1) cohorts, given their study line requests, and 2) elective course classes,

given the requests for these. This problem has been modeled as a bipartite network model and solved using a state-of-the-art MIP solver. A sequential solution approach produces solutions with an average of 16.4% optimality gap. Furthermore, the sequential approach gives solutions in one minute that are only 0.5% worse on average than the best solutions found within four hours.

Meeting Planning Problem: A new educational planning problem has been presented in this thesis, the Meeting Planning Problem. The problem is of assigning interrelated meetings to a timetable. There exist two different meeting planning problems at the Danish high schools; the Parental Consultation Timetabling Problem (PCTP) and the Supervisor Consultation Timetabling Problem (SCTP). One MIP model containing both consultation types has been created and solved using ALNS. It has been possible to provide solutions with an average of 2.31% from optimum for PCTP and 1.26% for SCTP.

A more generalized model of the Meeting Planning Problem is introduced and a Branch-and-Price algorithm has been created as the solution method and tested on the consultation cases.

Operational Research Techniques: Various operational research techniques have been used during the process of this thesis. Firstly, conceptual and mathematical models have been created for all the planning problems of this thesis. Secondly, different exact and heuristic solution methods have been applied with success.

Adaptive Large Neighborhood Search (ALNS) has been applied in all three research directions of this thesis. ALNS is a type of hyper-heuristic, where multiple insertion and removal heuristics are applied. The method is widely used on transportation problems, and through this thesis it is shown that it is an efficient method for solving educational planning problems. In those three cases where ALNS has been used, the results have been competitive and provided solutions close to optimum.

The exact solution methods applied throughout this thesis have been some simple methods, such as direct or sequential methods using a state-of-the-art MIP solver, and some more advanced approaches, such as decomposition and Branch-and-Price. Branch-and-Price has been applied for two different cases in this thesis; one using Dantzig-Wolfe and one using column generation. In both cases the methods have provided solutions close to optimum. A relatively new interesting branching technique, Explicit Constraint Branching, has been applied to enhance the solution process.

10.2 Practical Contribution

From a practical point, the Ph.D. project has resulted in the following contributions in chronological order:

- **Consultation Timetabling Problem:** An Adaptive Large Neighborhood Search algorithm has been implemented in Lectio and made it possible for the users to solve both the Parental Consultation Timetabling Problem and the Supervisor Consultation Timetabling Problem. The algorithm was released in May 2011.

- **Elective Course Student Sectioning:** An Adaptive Large Neighborhood Search algorithm was also the solution method chosen for the Elective Course Student Sectioning problem. A complete rewrite of the problem and the user interface has made it easier for the users to access the problem. The algorithm was released in January 2012

- **High School Timetabling:** The research on High School Timetabling has not resulted in a practical implementation, but contributed in a more indirect manner. MaCom A/S has used the partitioning of the Third International Timetabling Competition in 2011 and the resulting third place to create some positive publicity. Using various media, MaCom A/S has stated the solution methods in Lectio have been developed by the finalists of the *"World*

Cup of High School Timetabling" . This has, among other things, resulted in collaboration with Aarhus University concerning their timetabling problems.

- **High School Student Sectioning:** The research on High School Student Sectioning has not yet resulted in practical implementation. However, it is proved that the problem can be modeled and solved using operational research techniques instead of manual work, and this has increased the interest from both MaCom A/S and from the high schools.

Moreover, a generic Adaptive Large Neighborhood Search framework has been implemented in Lectio. This makes it easier to implement new heuristics for future planning problems.

10.3 Discussion and Directions of Future Research

This thesis presents a number of different educational planning problems which can all be investigated further. There are many interesting directions of future research for each individual research subject, and on educational planning problems in general. A few of these are presented in this section.

The amount of research papers on High School Timetabling has increased significantly the past decade, which is partly due to the XHSTT format. Many of the articles are concerned with heuristic solution methods to improve best known solutions. With the created MIP model of XHSTT, we have opened the area for more research using exact methods and creating lower bounds for the instances of XHSTT. Some of the instances are so large and complex that more research is needed to make it possible to establish bounds, and hence be able to measure the quality of the solutions.

Two Student Sectioning problems were presented and solved in this thesis. Elective Course Student Sectioning at the Danish high schools must be considered a closed research subject, as we are able to create solutions with an average gap of 1% from the optimum. For the High School Student Sectioning problem, we were only able to provide solutions with an average of 16.4% optimality gap and more research is needed to minimize this.

The research on the Meeting Planning Problem has presented a new research subject of educational planning problems. The Consultation Timetabling must also be considered a closed research chapter as the results are quite close to optimum. However, the General Meeting Planning Problem and the methods used, could be adapted to other similar meeting planning problems, such as Examination Timetabling.

The severity of the problems are varying. All the considered problems of this thesis contains some form of symmetry, which can cause problems with closing the optimality cap. However, for some of the problems this gap has been relatively small, and hence a minor detail. The most difficult problem of this thesis to solve, is the High School Timetabling problem of the XHSTT-format. In addition a large quantity of symmetry, the XHSTT contains some rather complex constraints, which had made it very difficult to obtain useful results for some of the instances.

One of the current trends of solution methods for educational planning problems, and operational research in general, is hyper-heuristics. A hyper-heuristic method has the advantage of combining multiple heuristics and hence benefit from the different techniques. It is likely that hyper-heuristics are going to be more prevalent in operational research. In this thesis ALNS has proved to be an efficient solution method for timetabling problems. By applying multiple removal and insertion heuristics, which is the essence of ALNS, to timetabling problems, it makes it possible to search the entire solution space. For achieving the best results with an ALNS algorithm, it is necessary to create several different removal and insertion heuristics. Furthermore, tuning the parameters of ALNS is crucial, as these are used for selecting heuristics and for acceptance of the solutions.

ALNS is a strong hyper-heuristic and is definitely a solution approach to consider when solving all kinds of planning problems, not just timetabling and routing problems.

For future research, it could be interesting to incorporated some symmetry breaking methods within an ALNS framework. Another possible future research matter could be to use more exact methods in a hyper-heuristic framework, by further developing the concept of hyper-heuristics and matheuristics.

Seen from a broader perspective, the most challenging task within Educational Timetabling is still the issue of closing the gap between theory and practice. One of the reasons for this is the lack of universal formats and broad benchmark data. High School Timetabling will benefited hugely from the introduction of the XHSTT format, which contains good varied benchmark data instances from all over the world. The task of creating a universal format for a given planning problem is a very complex and time-consuming task. However, this is essential if the goal is to minimize the gap between theory and practice. Other educational planning problems such as Universal Course Timetabling and Examination Timetabling do have benchmark data. However, these are often only concentrated on a few universities. Student Sectioning does not have any benchmark data at all.

10.4 Final Remarks

The work of this thesis highlights some of the educational planning problems, and the relevance and importance of developing optimization methods in a decision support framework.

This thesis has investigated high school planning problems and methods for solving these in terms of an operational research professional. The field of educational timetabling is an interesting field within operational research, and there is still plenty of research to be done and insights to gain on the subject, especially, on the task of closing the famous gap between theory and practice.